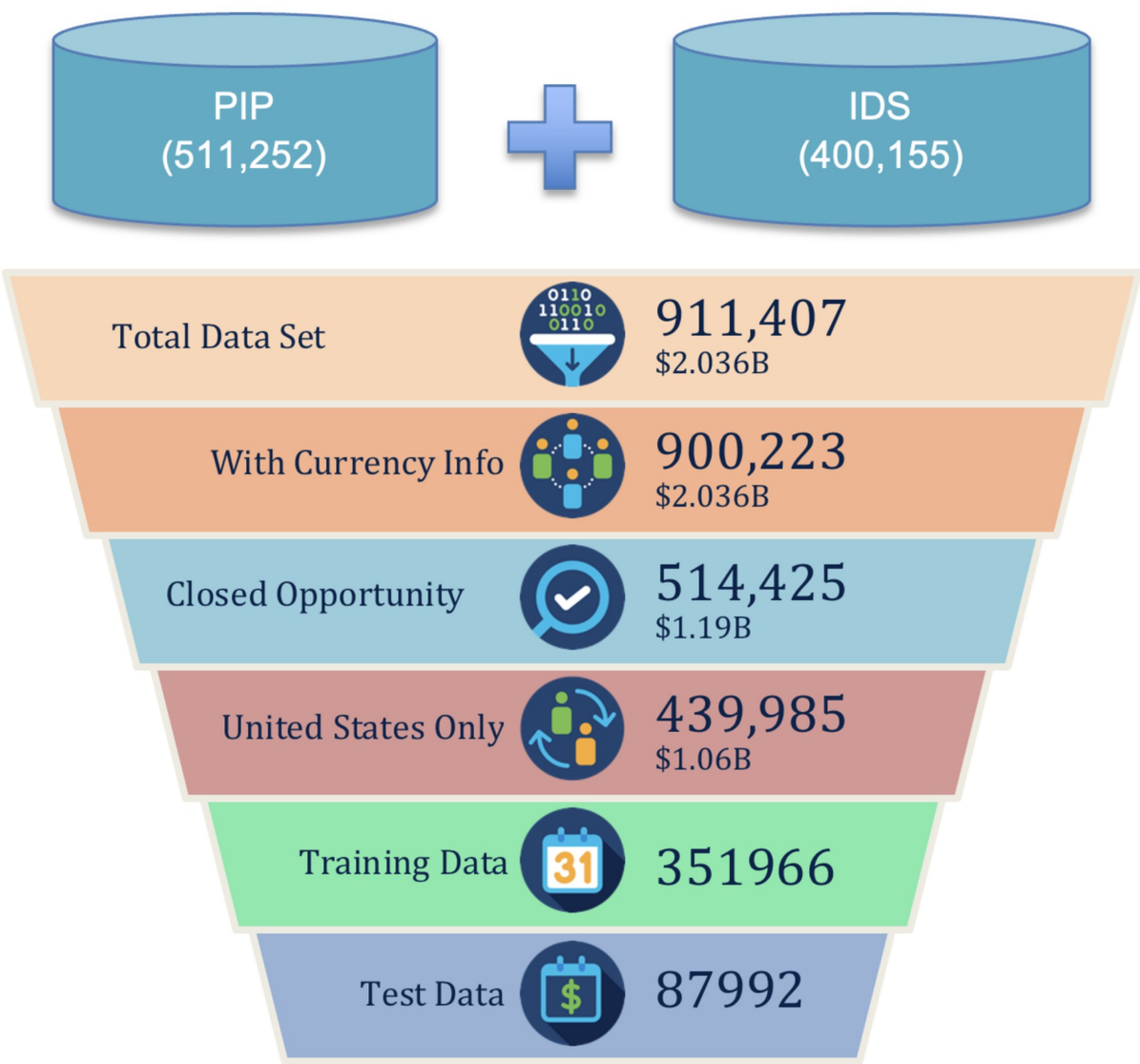
In [1]:
```
#Import data and libraries
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from datetime import date, timedelta
import matplotlib.ticker as mtick
import matplotlib.ticker as ticker
import scipy.stats as stats
from sklearn.preprocessing   import StandardScaler
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
import missingno as msno
from sklearn.cluster import KMeans

from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
import numpy as np
from sklearn.datasets import load_boston
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFECV
import matplotlib.pyplot as plt
# pip install hvplot install if don't have
from sklearn import metrics
from sklearn.model_selection import cross_val_score
import hvplot.pandas
#Encoder the categorical variables for nomial features
from sklearn.preprocessing import OneHotEncoder
import category_encoders as ce
```

In [2]:
```
#Import dataset
alldata = pd.read_csv('Guideline_Pricing_Full_Data.csv')
print("Dataset shape:\n",alldata.shape,"\n")
```

```
Dataset shape:
 (911407, 100)
```



Initial EDA

In [3]:
```
alldata.info() #get overall view of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 911407 entries, 0 to 911406
Data columns (total 100 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Account Manager                 814511 non-null  object
 1   Amendment Id                    77480 non-null   object
 2   Businessunit                    911407 non-null  object
 3   Charge Type                     900223 non-null  object
 4   City                            911350 non-null  object
 5   Close Reason                    496740 non-null  object
 6   Commitment Type                 911271 non-null  object
 7   Contract Id                     158149 non-null  object
 8   Contract Scenario Id            110060 non-null  float64
 9   Contracted Flag                 911407 non-null  object
 10  Country                         911399 non-null  object
 11  Country (group)                 911407 non-null  object
 12  Currency                        900223 non-null  object
 13  Discount Level                  241866 non-null  object
 14  Drv Port Speed                  911393 non-null  object
 15  Drv Port Type                   911399 non-null  object
 16  Feat Act Code Type              911407 non-null  object
 17  Feature Name                    911407 non-null  object
 18  First Scenario Date             511252 non-null  object
 19  First Scenario Dt               511252 non-null  object
 20  GEO                             911366 non-null  object
 21  Li Jpa                          911366 non-null  float64
 22  Nasp Type                       911407 non-null  object
 23  Naspid                          911366 non-null  object
 24  New Order Flag                  911407 non-null  object
 25  Next Order Flag                 911407 non-null  object
 26  Opportunity Type                898216 non-null  object
 27  Opty Id                         911366 non-null  object
 28  Pcm Touched                     911407 non-null  object
 29  Pip Node City                   217574 non-null  object
 30  Port Type                       502711 non-null  object
 31  Postal Cd                       907452 non-null  object
 32  Product Name                    911407 non-null  object
 33  Product Name (group)            911407 non-null  object
 34  Promo Id                        911366 non-null  object
 35  Promo Pcm Analyst Floor         397512 non-null  float64
 36  Promo Pcm Vp Floor              397512 non-null  float64
 37  Promo Txn Best Sales Price      0 non-null       float64
 38  Quote Id                        911407 non-null  int64
 39  Region Name                     911407 non-null  object
 40  Saleschannel                    911407 non-null  object
 41  Scenario Category               911407 non-null  object
 42  Scenario Id                     911407 non-null  int64
 43  Scenario Type                   911407 non-null  object
 44  Scenario Version Status         511252 non-null  object
 45  Sheet                           911407 non-null  object
 46  Site Id                         911407 non-null  object
 47  Sp Acc Provider                 898632 non-null  object
 48  Sp Acc Tech                     898632 non-null  object
 49  Sp Cust Hand Off Type           898632 non-null  object
 50  Sp Dbw                          508166 non-null  object
 51  Sp Internet Traffic             132 non-null     object
 52  Sp Max Bandwidth                498723 non-null  object
 53  Sp Port Speed                   511246 non-null  object
 54  Sp Port Type                    511252 non-null  object
 55  Sp Vzb Lit Status               898601 non-null  object
 56  SP_IDS_PRICEPLAN                400147 non-null  object
 57  SP_IDS_PRT_SPEED                400147 non-null  object
 58  SP_IDS_PRTTYP                   400147 non-null  object
 59  Stage                           897723 non-null  object
 60  Stage (group)                   911407 non-null  object
 61  State                           895726 non-null  object
 62  Table Name                      911407 non-null  object
 63  Term                            911366 non-null  float64
 64  Transaction Type                911407 non-null  object
 65  UNIQUE_PRODUCT_IDENTIFIER       911393 non-null  object
 66  Unitdisc                        911366 non-null  object
 67  Vertical Reporting Dsc          911406 non-null  object
 68  # of Contracts                  911407 non-null  int64
 69  # of NASP Customer              911407 non-null  int64
 70  # of Opty                        911407 non-null  int64
 71  Commitment Amount               911407 non-null  int64
 72  Li Bi Nre Direct Cost           911366 non-null  float64
 73  Li Bi Nre Discount              911366 non-null  float64
 74  Li Bi Nre List Price            911366 non-null  float64
 75  Li Bi Nre Net Price             911366 non-null  float64
 76  Li Bi Nre Promo                 911366 non-null  float64
 77  Li Bi Nre Shared Cost           911366 non-null  float64
 78  Li Bi Rec Direct Cost           911366 non-null  float64
 79  Li Bi Rec Discount              911366 non-null  float64
 80  Li Bi Rec List Price            911366 non-null  float64
 81  Li Bi Rec Net Price             911366 non-null  float64
 82  Li Bi Rec Promo                 911366 non-null  float64
 83  Li Bi Rec Shared Cost           911366 non-null  float64
 84  Li Bi Term Total Direct Cost    911366 non-null  float64
 85  Li Bi Term Total Net Price      911366 non-null  float64
 86  Li Bi Total Shared Cost         911366 non-null  float64
 87  Li Dd Rec Net Price             903861 non-null  float64
 88  Nasp Annual Tbr                 858501 non-null  float64
 89  Pcm Analyst Floor Rate          900223 non-null  float64
 90  Pcm Vp Floor Rate               900223 non-null  float64
 91  Price                           900223 non-null  float64
 92  Pstlzd Rate Usd Mrc             911366 non-null  float64
 93  Pstlzd Rate Usd Nrc             911366 non-null  float64
 94  Sales Vp Floor Rtb              502711 non-null  float64
 95  Sales Vp Floor Txn              502711 non-null  float64
 96  SALES_VP_FLOOR_TXN_RTB          397512 non-null  float64
 97  Scenario Version                911407 non-null  int64
```

```
 98  Selected Measure              900223 non-null  float64
 99  Sp Port Qty                   511219 non-null  float64
dtypes: float64(33), int64(7), object(60)
memory usage: 695.3+ MB
```

In [4]:
```
alldata['Table Name'].value_counts() #look at two products records
```

Out[4]:
```
MPLS_DATA    511252
IDS_DATA     400155
Name: Table Name, dtype: int64
```

In [5]:
```
alldata['Currency'].value_counts() #Get a count for currency info
```

Out[5]:
```
USD    900223
Name: Currency, dtype: int64
```

In [6]:
```
filtered_df = alldata[alldata['Currency'].notnull()].reset_index(drop=True) #filter the dataset with currency info
```

In [7]:
```
filtered_df['Country'].value_counts()  #Get a count for country info
```

Out[7]:
```
UNITED STATES     775989
CANADA             22005
UNITED KINGDOM     21606
GERMANY            10969
FRANCE              8739
                   ...
JORDAN                17
PERU                  11
SRI LANKA             10
ICELAND                9
TURKEY                 2
Name: Country, Length: 64, dtype: int64
```

In [8]:
```
#filter the dataset with UNITED STATES in country as this project main focus
filtered_df = filtered_df.loc[filtered_df['Country'] == 'UNITED STATES'].reset_index(drop=True)
```

In [9]:
```
filtered_df['Stage (group)'].value_counts() #look at the closed opportunities
```

Out[9]:
```
5 Closed Disqualified, 5 Closed Lost, 5 Closed Won    439985
Null, 0 Identify, 1 Qualify and 3 more                336004
Name: Stage (group), dtype: int64
```

In [10]:
```
filtered_df['Stage (group)'].unique() #look at the closed opportunities
```

Out[10]:
```
array(['5 Closed Disqualified, 5 Closed Lost, 5 Closed Won',
       'Null, 0 Identify, 1 Qualify and 3 more'], dtype=object)
```

In [11]:
```
#Transforming some numerical variables that are  categorical
filtered_df['Scenario Version'] = filtered_df['Scenario Version'].apply(str)
```

Open_Opportunity for modeling, we will use Close_Opportunity for predicting point price

In [12]:
```
Open_Opportunity = filtered_df.loc[filtered_df['Stage (group)'] != '5 Closed Disqualified, 5 Closed Lost, 5 Closed Won'].rese
```

In [13]:
```
#filter the dataset with closed opportunities only
Close_Opportunity = filtered_df.loc[filtered_df['Stage (group)'] == '5 Closed Disqualified, 5 Closed Lost, 5 Closed Won'].res
```

In [14]:
```
print("\nThe train data size for Open_Opportunity is : {} ".format(Open_Opportunity.shape))
print("The predict data size for Close_Opportunity is : {} ".format(Close_Opportunity.shape))
```

```
The train data size for Open_Opportunity is : (336004, 100)
The predict data size for Close_Opportunity is : (439985, 100)
```

In [15]:
```
filtered_df.info() #working data brife info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 775989 entries, 0 to 775988
Data columns (total 100 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Account Manager                 701165 non-null  object
 1   Amendment Id                    64671 non-null   object
 2   Businessunit                    775989 non-null  object
 3   Charge Type                     775989 non-null  object
 4   City                            775989 non-null  object
 5   Close Reason                    421957 non-null  object
 6   Commitment Type                 775959 non-null  object
 7   Contract Id                     138267 non-null  object
 8   Contract Scenario Id            92460 non-null   float64
 9   Contracted Flag                 775989 non-null  object
 10  Country                         775989 non-null  object
 11  Country (group)                 775989 non-null  object
 12  Currency                        775989 non-null  object
 13  Discount Level                  180100 non-null  object
 14  Drv Port Speed                  775989 non-null  object
 15  Drv Port Type                   775989 non-null  object
 16  Feat Act Code Type              775989 non-null  object
 17  Feature Name                    775989 non-null  object
 18  First Scenario Date             444576 non-null  object
 19  First Scenario Dt               444576 non-null  object
 20  GEO                             775989 non-null  object
 21  Li Jpa                          775989 non-null  float64
 22  Nasp Type                       775989 non-null  object
 23  Naspid                          775989 non-null  object
 24  New Order Flag                  775989 non-null  object
 25  Next Order Flag                 775989 non-null  object
 26  Opportunity Type                763417 non-null  object
 27  Opty Id                         775989 non-null  object
 28  Pcm Touched                     775989 non-null  object
 29  Pip Node City                   217574 non-null  object
 30  Port Type                       444576 non-null  object
 31  Postal Cd                       775954 non-null  object
 32  Product Name                    775989 non-null  object
 33  Product Name (group)            775989 non-null  object
 34  Promo Id                        775989 non-null  object
 35  Promo Pcm Analyst Floor         331413 non-null  float64
 36  Promo Pcm Vp Floor              331413 non-null  float64
 37  Promo Txn Best Sales Price      0 non-null       float64
 38  Quote Id                        775989 non-null  int64
 39  Region Name                     775989 non-null  object
 40  Saleschannel                    775989 non-null  object
 41  Scenario Category               775989 non-null  object
 42  Scenario Id                     775989 non-null  int64
 43  Scenario Type                   775989 non-null  object
 44  Scenario Version Status         444576 non-null  object
 45  Sheet                           775989 non-null  object
 46  Site Id                         775989 non-null  object
 47  Sp Acc Provider                 765789 non-null  object
 48  Sp Acc Tech                     765789 non-null  object
 49  Sp Cust Hand Off Type           765789 non-null  object
 50  Sp Dbw                          444576 non-null  object
 51  Sp Internet Traffic             132 non-null     object
 52  Sp Max Bandwidth                441110 non-null  object
 53  Sp Port Speed                   444576 non-null  object
 54  Sp Port Type                    444576 non-null  object
 55  Sp Vzb Lit Status               765764 non-null  object
 56  SP_IDS_PRICEPLAN                331413 non-null  object
 57  SP_IDS_PRT_SPEED                331413 non-null  object
 58  SP_IDS_PRTTYP                   331413 non-null  object
 59  Stage                           762973 non-null  object
 60  Stage (group)                   775989 non-null  object
 61  State                           775989 non-null  object
 62  Table Name                      775989 non-null  object
 63  Term                            775989 non-null  float64
 64  Transaction Type                775989 non-null  object
 65  UNIQUE_PRODUCT_IDENTIFIER       775989 non-null  object
 66  Unitdisc                        775989 non-null  object
 67  Vertical Reporting Dsc          775988 non-null  object
 68  # of Contracts                  775989 non-null  int64
 69  # of NASP Customer              775989 non-null  int64
 70  # of Opty                        775989 non-null  int64
 71  Commitment Amount               775989 non-null  int64
 72  Li Bi Nre Direct Cost           775989 non-null  float64
 73  Li Bi Nre Discount              775989 non-null  float64
 74  Li Bi Nre List Price            775989 non-null  float64
 75  Li Bi Nre Net Price             775989 non-null  float64
 76  Li Bi Nre Promo                 775989 non-null  float64
 77  Li Bi Nre Shared Cost           775989 non-null  float64
 78  Li Bi Rec Direct Cost           775989 non-null  float64
 79  Li Bi Rec Discount              775989 non-null  float64
 80  Li Bi Rec List Price            775989 non-null  float64
 81  Li Bi Rec Net Price             775989 non-null  float64
 82  Li Bi Rec Promo                 775989 non-null  float64
 83  Li Bi Rec Shared Cost           775989 non-null  float64
 84  Li Bi Term Total Direct Cost    775989 non-null  float64
 85  Li Bi Term Total Net Price      775989 non-null  float64
 86  Li Bi Total Shared Cost         775989 non-null  float64
 87  Li Dd Rec Net Price             769920 non-null  float64
 88  Nasp Annual Tbr                 730772 non-null  float64
 89  Pcm Analyst Floor Rate          775989 non-null  float64
 90  Pcm Vp Floor Rate               775989 non-null  float64
 91  Price                           775989 non-null  float64
 92  Pstlzd Rate Usd Mrc             775989 non-null  float64
 93  Pstlzd Rate Usd Nrc             775989 non-null  float64
 94  Sales Vp Floor Rtb              444576 non-null  float64
 95  Sales Vp Floor Txn              444576 non-null  float64
 96  SALES_VP_FLOOR_TXN_RTB          331413 non-null  float64
 97  Scenario Version                775989 non-null  object
```

```
 98  Selected Measure              775989 non-null  float64
 99  Sp Port Qty                   444576 non-null  float64
dtypes: float64(33), int64(6), object(61)
memory usage: 592.0+ MB
```

In [16]:
```
Open_Opportunity.dtypes.value_counts(normalize = True) #check the % of datatypes
```

Out[16]:
```
object     0.61
float64    0.33
int64      0.06
dtype: float64
```

In [17]:
```
#change the id data type to string
filtered_df[["Quote Id", "Scenario Id","Li Jpa" , "Contract Scenario Id"]] = filtered_df[["Quote Id", "Scenario Id","Li Jpa"
```

In [18]:
```
filtered_df.describe().transpose().applymap("{0:,.2f}".format) #get a summary for numberical variables
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Promo Pcm Analyst Floor	331,413.00	265.35	326.33	0.00	153.45	186.75	221.40	2,855.70
Promo Pcm Vp Floor	331,413.00	213.32	307.39	0.00	119.35	145.25	172.20	2,856.00
Promo Txn Best Sales Price	0.00	nan	nan	nan	nan	nan	nan	nan
Term	775,989.00	33.95	12.72	0.00	36.00	36.00	36.00	132.00
# of Contracts	775,989.00	0.18	0.38	0.00	0.00	0.00	0.00	1.00
# of NASP Customer	775,989.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
# of Opty	775,989.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
Commitment Amount	775,989.00	278,580.18	4,703,408.81	0.00	0.00	10,000.00	10,000.00	500,000,000.00
Li Bi Nre Direct Cost	775,989.00	314.38	196.42	0.00	85.90	311.25	524.19	1,742.64
Li Bi Nre Discount	775,989.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Li Bi Nre List Price	775,989.00	49.79	175.57	0.00	0.00	0.00	0.00	5,000.00
Li Bi Nre Net Price	775,989.00	1.54	35.42	-1,000.00	0.00	0.00	0.00	5,000.00
Li Bi Nre Promo	775,989.00	-48.35	172.78	-1,000.00	0.00	0.00	0.00	0.00
Li Bi Nre Shared Cost	775,989.00	0.09	1.93	-29.46	0.00	0.00	0.00	245.50
Li Bi Rec Direct Cost	775,989.00	78.92	201.82	0.00	27.16	32.14	58.30	5,261.07
Li Bi Rec Discount	775,989.00	-145.49	1,008.67	-143,500.00	0.00	0.00	0.00	550.36
Li Bi Rec List Price	775,989.00	718.49	1,969.95	0.00	147.60	346.00	775.00	205,000.00
Li Bi Rec Net Price	775,989.00	499.51	1,371.04	-64,457.20	133.00	221.50	465.00	205,000.00
Li Bi Rec Promo	775,989.00	-0.15	5.27	-2,636.00	0.00	0.00	0.00	0.00
Li Bi Rec Shared Cost	775,989.00	360.42	472.65	0.00	82.53	139.48	553.20	14,013.23
Li Bi Term Total Direct Cost	775,989.00	2,893.37	6,949.82	0.00	1,173.82	1,520.85	2,321.24	272,815.65
Li Bi Term Total Net Price	775,989.00	16,079.83	50,565.98	-773,486.40	3,939.84	7,207.20	15,000.00	9,840,000.00
Li Bi Total Shared Cost	775,989.00	11,768.74	16,477.19	0.00	2,799.84	4,554.72	13,836.24	556,523.04
Li Dd Rec Net Price	769,920.00	500.62	1,380.61	0.00	133.00	221.50	466.00	205,000.00
Nasp Annual Tbr	730,772.00	16,124,540.55	26,328,555.45	-2,547,021.82	375,757.65	4,318,568.91	17,787,013.30	933,725,658.10
Pcm Analyst Floor Rate	775,989.00	959.61	3,127.74	60.00	180.00	356.40	572.00	28,978.00
Pcm Vp Floor Rate	775,989.00	726.28	2,346.10	45.00	136.50	275.00	429.00	21,734.00
Price	775,989.00	2,333.31	7,818.33	150.00	450.00	800.00	1,431.00	72,446.00
Pstlzd Rate Usd Mrc	775,989.00	29.78	194.05	0.00	0.00	0.00	0.00	24,202.00
Pstlzd Rate Usd Nrc	775,989.00	0.12	7.60	0.00	0.00	0.00	0.00	640.00
Sales Vp Floor Rtb	444,576.00	1,289.51	4,075.97	60.00	180.00	313.00	572.00	28,978.00
Sales Vp Floor Txn	444,576.00	1,612.10	5,094.99	75.00	225.00	391.00	716.00	36,223.00
SALES_VP_FLOOR_TXN_RTB	331,413.00	592.14	602.40	130.00	202.80	464.36	540.80	4,940.00
Selected Measure	775,989.00	2,333.31	7,818.33	150.00	450.00	800.00	1,431.00	72,446.00
Sp Port Qty	444,576.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00

In [19]:
```
#get the descriptions for all categorical variables
for i in filtered_df.select_dtypes(include=['object']).columns:
    print(i)
    print('Count of unique varuables:', len(filtered_df[i].unique()))
    print('Top 3 most common variables in ', i)
    print(filtered_df[i].value_counts().head(3),'\n')
```

```
Account Manager
Count of unique varuables: 1276
Top 3 most common variables in  Account Manager
Humphrey, Gary      36605
Haydon, Sean        33879
Mayhew, Alexia L    29080
Name: Account Manager, dtype: int64

Amendment Id
Count of unique varuables: 14540
Top 3 most common variables in  Amendment Id
G24963-01    420
H45517-00    354
G73721-01    335
Name: Amendment Id, dtype: int64

Businessunit
Count of unique varuables: 3
Top 3 most common variables in  Businessunit
Global Enterprise    602435
Public Sector        106566
Business Markets      66988
Name: Businessunit, dtype: int64

Charge Type
Count of unique varuables: 1
Top 3 most common variables in  Charge Type
Recurring    775989
Name: Charge Type, dtype: int64

City
Count of unique varuables: 11441
Top 3 most common variables in  City
NEW YORK    12307
ATLANTA      9695
HOUSTON      7978
Name: City, dtype: int64

Close Reason
Count of unique varuables: 27
Top 3 most common variables in  Close Reason
Won: Price / Contract Terms    179937
DQ: Oppty Withdrawal            69493
Won: Product Offering           62212
Name: Close Reason, dtype: int64

Commitment Type
Count of unique varuables: 7
Top 3 most common variables in  Commitment Type
AVC                               770981
TVC                                 2895
Tiered Annual Volume Commitment     1819
Name: Commitment Type, dtype: int64

Contract Id
Count of unique varuables: 7032
Top 3 most common variables in  Contract Id
G88530-03    15919
301633        9030
140217        8122
Name: Contract Id, dtype: int64

Contract Scenario Id
Count of unique varuables: 6795
Top 3 most common variables in  Contract Scenario Id
nan            683529
130344645.0     15919
3100576.0        9030
Name: Contract Scenario Id, dtype: int64

Contracted Flag
Count of unique varuables: 2
Top 3 most common variables in  Contracted Flag
N    635798
Y    140191
Name: Contracted Flag, dtype: int64

Country
Count of unique varuables: 1
Top 3 most common variables in  Country
UNITED STATES    775989
Name: Country, dtype: int64

Country (group)
Count of unique varuables: 1
Top 3 most common variables in  Country (group)
UNITED STATES    775989
Name: Country (group), dtype: int64

Currency
Count of unique varuables: 1
Top 3 most common variables in  Currency
USD    775989
Name: Currency, dtype: int64

Discount Level
Count of unique varuables: 11
Top 3 most common variables in  Discount Level
PCM    88062
VP     57725
FLO    25674
Name: Discount Level, dtype: int64
```

```
Drv Port Speed
Count of unique varuables: 48
Top 3 most common variables in  Drv Port Speed
10 Mbps     217845
100 Mbps    106683
50 Mbps      81491
Name: Drv Port Speed, dtype: int64

Drv Port Type
Count of unique varuables: 2
Top 3 most common variables in  Drv Port Type
Ethernet    770184
TDM           5805
Name: Drv Port Type, dtype: int64

Feat Act Code Type
Count of unique varuables: 4
Top 3 most common variables in  Feat Act Code Type
ADD         724797
CHANGE       31762
EXISTING     19026
Name: Feat Act Code Type, dtype: int64

Feature Name
Count of unique varuables: 2
Top 3 most common variables in  Feature Name
PIP Port                  444576
Internet Dedicated Port    331413
Name: Feature Name, dtype: int64

First Scenario Date
Count of unique varuables: 531
Top 3 most common variables in  First Scenario Date
11/20/2020    27941
9/18/2020     13701
9/11/2020     13020
Name: First Scenario Date, dtype: int64

First Scenario Dt
Count of unique varuables: 30060
Top 3 most common variables in  First Scenario Dt
08-DEC-20 02.16.17.803000000 PM    6110
13-SEP-21 07.56.18.265000000 AM    4078
22-SEP-20 06.20.53.831000000 PM    2230
Name: First Scenario Dt, dtype: int64

GEO
Count of unique varuables: 1
Top 3 most common variables in  GEO
USA    775989
Name: GEO, dtype: int64

Li Jpa
Count of unique varuables: 557775
Top 3 most common variables in  Li Jpa
9671301368.0    4
8675231048.0    4
9681449783.0    4
Name: Li Jpa, dtype: int64

Nasp Type
Count of unique varuables: 159
Top 3 most common variables in  Nasp Type
E3 - US          99014
E2 - US          90424
E1 VWAG2 - US    89532
Name: Nasp Type, dtype: int64

Naspid
Count of unique varuables: 18224
Top 3 most common variables in  Naspid
YYY40WFBKXXXX    55048
YYY10AMZSXXXX    35318
YYY10ETNAXXXX    30307
Name: Naspid, dtype: int64

New Order Flag
Count of unique varuables: 2
Top 3 most common variables in  New Order Flag
N    679390
Y     96599
Name: New Order Flag, dtype: int64

Next Order Flag
Count of unique varuables: 2
Top 3 most common variables in  Next Order Flag
N    720538
Y     55451
Name: Next Order Flag, dtype: int64

Opportunity Type
Count of unique varuables: 17
Top 3 most common variables in  Opportunity Type
New Business               585776
Add to Existing/Upgrade     56712
Renewal Only                31834
Name: Opportunity Type, dtype: int64

Opty Id
Count of unique varuables: 39355
Top 3 most common variables in  Opty Id
YYYO-2443598XXXX     41568
YYYO-2868555XXXX     34058
```

```
YYYO2-0049125XXXX    29078
Name: Opty Id, dtype: int64

Pcm Touched
Count of unique varuables: 2
Top 3 most common variables in  Pcm Touched
N    516595
Y    259394
Name: Pcm Touched, dtype: int64

Pip Node City
Count of unique varuables: 2
Top 3 most common variables in  Pip Node City
Anchorage    217574
Name: Pip Node City, dtype: int64

Port Type
Count of unique varuables: 3
Top 3 most common variables in  Port Type
Ethernet    439866
TDM           4710
Name: Port Type, dtype: int64

Postal Cd
Count of unique varuables: 193631
Top 3 most common variables in  Postal Cd
20147-6205    878
75207-3134    786
95119-1242    561
Name: Postal Cd, dtype: int64

Product Name
Count of unique varuables: 2
Top 3 most common variables in  Product Name
Private IP (PIP)               444576
Internet Dedicated Services    331413
Name: Product Name, dtype: int64

Product Name (group)
Count of unique varuables: 1
Top 3 most common variables in  Product Name (group)
Internet Dedicated Services & Private IP (PIP)    775989
Name: Product Name (group), dtype: int64

Promo Id
Count of unique varuables: 145
Top 3 most common variables in  Promo Id
-99              570002
100011            40293
100007;100165     21953
Name: Promo Id, dtype: int64

Quote Id
Count of unique varuables: 71980
Top 3 most common variables in  Quote Id
204736551    15914
207174151     5356
204794265     3232
Name: Quote Id, dtype: int64

Region Name
Count of unique varuables: 24
Top 3 most common variables in  Region Name
PREMIER      213117
PRINCIPAL    144176
SIGNATURE    108222
Name: Region Name, dtype: int64

Saleschannel
Count of unique varuables: 2
Top 3 most common variables in  Saleschannel
Direct    742556
VPP        33433
Name: Saleschannel, dtype: int64

Scenario Category
Count of unique varuables: 2
Top 3 most common variables in  Scenario Category
TRANSACTION       580832
RTB_FULLCONFIG    195157
Name: Scenario Category, dtype: int64

Scenario Id
Count of unique varuables: 82962
Top 3 most common variables in  Scenario Id
130344645    15914
194259382     4078
197736218     2714
Name: Scenario Id, dtype: int64

Scenario Type
Count of unique varuables: 5
Top 3 most common variables in  Scenario Type
ILLUSTRATIVE      366990
PROPOSED          195157
QUOTE_TO_ORDER     92623
Name: Scenario Type, dtype: int64

Scenario Version Status
Count of unique varuables: 11
Top 3 most common variables in  Scenario Version Status
SALES_VALIDATED      250092
PRICEBOOK_CREATED    100473
```

```
PCM_APPROVED          27235
Name: Scenario Version Status, dtype: int64

Sheet
Count of unique varuables: 2
Top 3 most common variables in  Sheet
MPLS!DATA    444576
IDS!DATA     331413
Name: Sheet, dtype: int64

Site Id
Count of unique varuables: 272802
Top 3 most common variables in  Site Id
81933785C    324
81933783C    228
47262419C    192
Name: Site Id, dtype: int64

Sp Acc Provider
Count of unique varuables: 389
Top 3 most common variables in  Sp Acc Provider
VZT    174784
ATT    132842
BS      77585
Name: Sp Acc Provider, dtype: int64

Sp Acc Tech
Count of unique varuables: 4
Top 3 most common variables in  Sp Acc Tech
Switched Ethernet            709742
Ethernet over TDM or DWDM     50401
TDM (DSn/OCn/SDH)              5646
Name: Sp Acc Tech, dtype: int64

Sp Cust Hand Off Type
Count of unique varuables: 3
Top 3 most common variables in  Sp Cust Hand Off Type
Ethernet    760143
TDM           5646
Name: Sp Cust Hand Off Type, dtype: int64

Sp Dbw
Count of unique varuables: 4
Top 3 most common variables in  Sp Dbw
Yes          434242
Burstable      6872
No             3462
Name: Sp Dbw, dtype: int64

Sp Internet Traffic
Count of unique varuables: 2
Top 3 most common variables in  Sp Internet Traffic
Internet Traffic - Converged IP    132
Name: Sp Internet Traffic, dtype: int64

Sp Max Bandwidth
Count of unique varuables: 41
Top 3 most common variables in  Sp Max Bandwidth
10 Mbps     112278
20 Mbps      80010
100 Mbps     62816
Name: Sp Max Bandwidth, dtype: int64

Sp Port Speed
Count of unique varuables: 48
Top 3 most common variables in  Sp Port Speed
10 Mbps    147565
2 Mbps      56659
20 Mbps     35531
Name: Sp Port Speed, dtype: int64

Sp Port Type
Count of unique varuables: 3
Top 3 most common variables in  Sp Port Type
Ethernet    439866
TDM           4710
Name: Sp Port Type, dtype: int64

Sp Vzb Lit Status
Count of unique varuables: 11
Top 3 most common variables in  Sp Vzb Lit Status
Off-Net               722846
On-Net                 30069
Colo - New Customer    12472
Name: Sp Vzb Lit Status, dtype: int64

SP_IDS_PRICEPLAN
Count of unique varuables: 4
Top 3 most common variables in  SP_IDS_PRICEPLAN
Tiered                  308093
Burstable Select         20878
Burstable Aggregation     2442
Name: SP_IDS_PRICEPLAN, dtype: int64

SP_IDS_PRT_SPEED
Count of unique varuables: 39
Top 3 most common variables in  SP_IDS_PRT_SPEED
100 Mbps    71502
10 Mbps     70280
50 Mbps     50810
Name: SP_IDS_PRT_SPEED, dtype: int64

SP_IDS_PRTTYP
```

```
Count of unique varuables: 3
Top 3 most common variables in  SP_IDS_PRTTYP
Ethernet    330318
TDM           1095
Name: SP_IDS_PRTTYP, dtype: int64

Stage
Count of unique varuables: 9
Top 3 most common variables in  Stage
5 Closed Won             259034
3 Propose                108004
5 Closed Disqualified    100608
Name: Stage, dtype: int64

Stage (group)
Count of unique varuables: 2
Top 3 most common variables in  Stage (group)
5 Closed Disqualified, 5 Closed Lost, 5 Closed Won    439985
Null, 0 Identify, 1 Qualify and 3 more                336004
Name: Stage (group), dtype: int64

State
Count of unique varuables: 51
Top 3 most common variables in  State
CA    70771
TX    61599
GA    59188
Name: State, dtype: int64

Table Name
Count of unique varuables: 2
Top 3 most common variables in  Table Name
MPLS_DATA    444576
IDS_DATA     331413
Name: Table Name, dtype: int64

Transaction Type
Count of unique varuables: 5
Top 3 most common variables in  Transaction Type
NEW          661213
AMENDMENT     67821
MACD          21383
Name: Transaction Type, dtype: int64

UNIQUE_PRODUCT_IDENTIFIER
Count of unique varuables: 95
Top 3 most common variables in  UNIQUE_PRODUCT_IDENTIFIER
Private IP (PIP)-10 Mbps-Ethernet               147553
Internet Dedicated Services-100 Mbps-Ethernet     71502
Internet Dedicated Services-10 Mbps-Ethernet      70280
Name: UNIQUE_PRODUCT_IDENTIFIER, dtype: int64

Unitdisc
Count of unique varuables: 9
Top 3 most common variables in  Unitdisc
PCM     517677
List    165367
VP       61314
Name: Unitdisc, dtype: int64

Vertical Reporting Dsc
Count of unique varuables: 23
Top 3 most common variables in  Vertical Reporting Dsc
FINANCE          177616
RETAIL           100940
PUBLIC SECTOR     92024
Name: Vertical Reporting Dsc, dtype: int64

Scenario Version
Count of unique varuables: 25
Top 3 most common variables in  Scenario Version
2    490929
1    139430
3     95544
Name: Scenario Version, dtype: int64
```

Feature extraction and categorize into 4 different data types

In [20]:
```
categorical_data = ['Businessunit','Discount Level','Close Reason',
                    'Commitment Type','Contracted Flag','Drv Port Speed','Sp Port Speed', 'Feat Act Code Type','Feature Na
                    'Nasp Type','New Order Flag','Next Order Flag','Opportunity Type','Pcm Touched','Product Name',
                    'Saleschannel','Scenario Category','Scenario Type','Scenario Version Status',
                    'Sp Acc Tech','Sp Cust Hand Off Type','Sp Dbw',
                    'Sp Vzb Lit Status','SP_IDS_PRICEPLAN', 'Stage (group)',
                    'Stage','Transaction Type','UNIQUE_PRODUCT_IDENTIFIER', 'Scenario Version',
                    'Unitdisc','Vertical Reporting Dsc']

numerical_data = ['Term',
                  'Commitment Amount',
                  'Li Bi Rec Direct Cost',
                  'Li Bi Rec List Price',
                  'Li Bi Rec Net Price',
                  'Price',
                  'Li Bi Rec Discount',
                  'Li Bi Rec Promo',
                  'Li Bi Rec Shared Cost',
                  'Li Bi Term Total Direct Cost',
                  'Li Bi Term Total Net Price',
                  'Li Bi Total Shared Cost',
                  'Li Dd Rec Net Price']

geo_data = ['City','State']

temporal_data = ['First Scenario Date']
```

Determined 4 different of data types, we will encode variables in categorical_data

In [21]:
```
#All variables for both Open_Opportunity and Closed_Opportunity
categorical_df = filtered_df[categorical_data]
numerical_df = filtered_df[numerical_data]
geo_df = filtered_df[geo_data]
temporal_df = filtered_df[temporal_data]
subset = pd.concat([categorical_df,numerical_df,geo_df,temporal_df], axis=1)


#All variables for Open_Opportunity only
categorical_df_open = Open_Opportunity[categorical_data]
numerical_df_open = Open_Opportunity[numerical_data]
geo_df_open = Open_Opportunity[geo_data]
temporal_df_open = Open_Opportunity[temporal_data]
subset_open = pd.concat([categorical_df_open,numerical_df_open,geo_df_open,temporal_df_open], axis=1)


#All variables for Closed_Opportunity only
categorical_df_close = Close_Opportunity[categorical_data]
numerical_df_close = Close_Opportunity[numerical_data]
geo_df_close = Close_Opportunity[geo_data]
temporal_df_close = Close_Opportunity[temporal_data]
subset_close = pd.concat([categorical_df_close,numerical_df_close,geo_df_close,temporal_df_close], axis=1)


print("\nThe train data size for subset is : {} ".format(subset.shape))
print("\nThe train data size for subset_open is : {} ".format(subset_open.shape))
print("\nThe predict data size for subset_close is : {} ".format(subset_close.shape))
```

```
The train data size for subset is : (775989, 47)

The train data size for subset_open is : (336004, 47)

The predict data size for subset_close is : (439985, 47)
```

In [22]:
```
subset.loc[:,subset.dtypes==np.object].head(3)
```

Out[22]:

	Businessunit	Discount Level	Close Reason	Commitment Type	Contracted Flag	Drv Port Speed	Sp Port Speed	Feat Act Code Type	Feature Name	Nasp Type	...	Stage (group)	Stage	Transaction Type	UN
0	Global Enterprise	NaN	Lost: Price	AVC	N	20 Mbps	NaN	ADD	Internet Dedicated Port	E2 VWAG2 - US	...	5 Closed Disqualified, 5 Closed Lost, 5 Closed...	5 Closed Lost	AMENDMENT	I
1	Global Enterprise	NaN	Lost: Price	AVC	N	20 Mbps	NaN	ADD	Internet Dedicated Port	E2 VWAG2 - US	...	5 Closed Disqualified, 5 Closed Lost, 5 Closed...	5 Closed Lost	AMENDMENT	I
2	Business Markets	FLO	NaN	AVC	N	5 Gbps	NaN	ADD	Internet Dedicated Port	MEDIUM BUSINESS	...	Null, 0 Identify, 1 Qualify and 3 more	0 Identify	NEW	

3 rows × 34 columns

In [23]:
```
subset.loc[:,subset.dtypes==np.number].head(3)
```

Out[23]:

	Term	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Discount	Li Bi Rec Promo	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	Li Dd Rec Net Price
0	36.0	36.36	306.4	306.40	766.0	0.00	0.0	199.58	1394.86	11030.40	7184.88	306.40
1	36.0	36.36	306.4	306.40	766.0	0.00	0.0	199.58	1394.86	11030.40	7184.88	306.40
2	60.0	1631.46	4712.0	2450.24	6000.0	-2261.76	0.0	2532.99	98198.85	137213.44	151979.40	2450.24

Data Processing

Outliers Analysis and Removal for Colsed_Opportunity data which we will use for training and modeling, leave untouched for Open_Opportunity data here

In [24]:
```
from scipy.stats import norm, skew
# Identifying Outliers with Skewness
skewed_feats = numerical_df_close.apply(lambda x: skew(x.dropna())).sort_values(ascending=False)
print("\nSkew in numerical features: \n")
skewness = pd.DataFrame({'Skew' :skewed_feats})
skewness
```

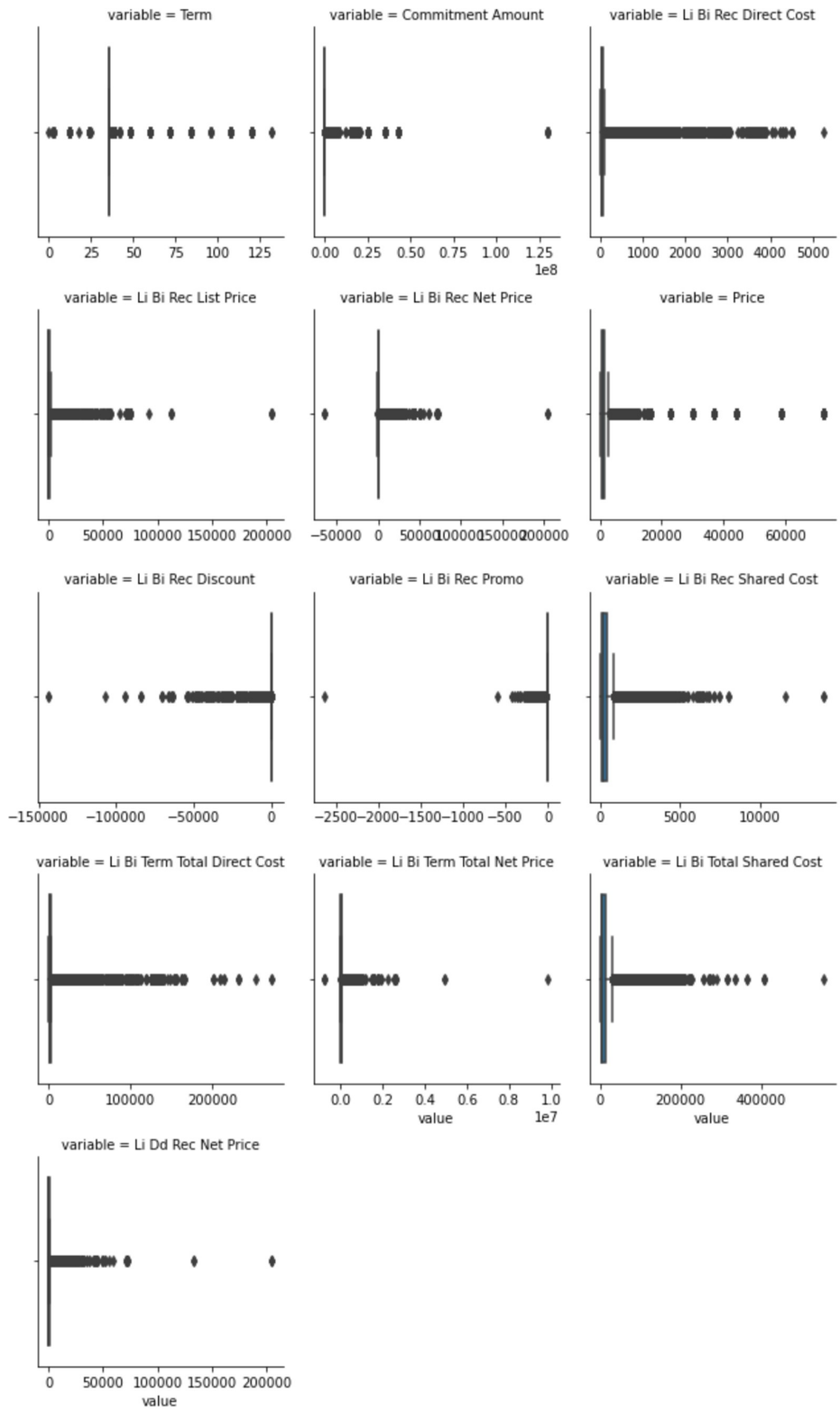
```
Skew in numerical features:
```

Out[24]:

	Skew
Li Bi Term Total Net Price	63.241700
Li Dd Rec Net Price	52.818564
Li Bi Rec Net Price	46.975124
Li Bi Rec List Price	31.430037
Commitment Amount	23.705285
Li Bi Term Total Direct Cost	12.630872
Li Bi Rec Direct Cost	10.626308
Price	7.512552
Li Bi Total Shared Cost	3.546178
Li Bi Rec Shared Cost	3.031489
Term	0.026957
Li Bi Rec Discount	-40.402596
Li Bi Rec Promo	-201.631073

In [25]:
```
g = sns.FacetGrid(pd.melt(subset_close[numerical_data]), col='variable',col_wrap=3,sharex=False, sharey=False)
g.map(sns.boxplot,'value')
```

Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x1ae65a7f580>
```



In [26]:
```
subset_close[numerical_data].describe()
```

Out[26]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Discount	Li Bi Rec Promo	Li Bi Rec Shared Cost	T
count	439985.000000	4.399850e+05	439985.000000	439985.000000	439985.000000	439985.000000	439985.000000	439985.000000	439985.000000	439
mean	32.559476	1.448579e+05	74.405435	689.228641	442.835689	2417.293385	-153.428670	-0.172574	350.091810	2
std	11.954231	2.081515e+06	182.774606	2008.305201	1217.048583	8100.709590	1181.888397	6.108154	458.636954	6
min	0.000000	0.000000e+00	0.000000	0.000000	-64457.200000	150.000000	-143500.000000	-2636.000000	0.000000	
25%	36.000000	0.000000e+00	27.160000	133.000000	119.000000	450.000000	0.000000	0.000000	116.700000	1
50%	36.000000	1.000000e+04	32.140000	312.000000	202.800000	893.000000	0.000000	0.000000	142.050000	1
75%	36.000000	1.000000e+04	54.330000	766.000000	425.000000	1431.000000	0.000000	0.000000	386.550000	2
max	132.000000	1.300000e+08	5261.070000	205000.000000	205000.000000	72446.000000	184.005000	0.000000	14013.230000	272

In [27]:
```
#We remove 2.5 percent percentile for both lower and upper boundary, identify these as outlier
Lower = subset_close.quantile(0.025)
Upper = subset_close.quantile(0.975)
IQR = Upper - Lower
print(IQR)
```

```
Term                              48.00
Commitment Amount              480000.00
Li Bi Rec Direct Cost             408.11
Li Bi Rec List Price             2939.00
Li Bi Rec Net Price              2440.00
Price                           11677.00
Li Bi Rec Discount               1190.00
Li Bi Rec Promo                     0.00
Li Bi Rec Shared Cost            1668.78
Li Bi Term Total Direct Cost    13276.29
Li Bi Term Total Net Price      80880.00
Li Bi Total Shared Cost         56805.12
Li Dd Rec Net Price              2440.00
dtype: float64
```

In [28]:
```
Lower - 1.5 * IQR
```

Out[28]:
```
Term                              -60.000
Commitment Amount              -720000.000
Li Bi Rec Direct Cost             -612.165
Li Bi Rec List Price             -4348.500
Li Bi Rec Net Price              -3600.000
Price                           -17290.500
Li Bi Rec Discount               -2975.000
Li Bi Rec Promo                      0.000
Li Bi Rec Shared Cost            -2503.170
Li Bi Term Total Direct Cost    -19914.435
Li Bi Term Total Net Price     -120600.000
Li Bi Total Shared Cost         -85207.680
Li Dd Rec Net Price              -3600.000
dtype: float64
```

In [29]:
```
Upper + 1.5 * IQR
```

Out[29]:
```
Term                              132.000
Commitment Amount              1200000.000
Li Bi Rec Direct Cost             1020.275
Li Bi Rec List Price              7407.500
Li Bi Rec Net Price               6160.000
Price                            29417.500
Li Bi Rec Discount                1785.000
Li Bi Rec Promo                      0.000
Li Bi Rec Shared Cost             4171.950
Li Bi Term Total Direct Cost     33190.725
Li Bi Term Total Net Price      202920.000
Li Bi Total Shared Cost         142012.800
Li Dd Rec Net Price               6160.000
dtype: float64
```

In [30]:
```
print(subset_close.shape)
subset_close_final = subset_close[~((subset_close < (Lower - 1.5 * IQR)) |(subset_close > (Upper + 1.5 * IQR))).any(axis=1)]
print(subset_close_final.shape)
```

```
(439985, 47)
(423111, 47)
```

We lost about 17970 record from closed opportunity data, about 4% of data after applying this technqiue.

In [31]:
```
# count of zero value in each column prior to the log
for col in numerical_data:
    print(col, subset_close_final[subset_close_final[col]==0].shape[0])
```

```
Term 0
Commitment Amount 138444
Li Bi Rec Direct Cost 29891
Li Bi Rec List Price 2449
Li Bi Rec Net Price 2724
Price 0
Li Bi Rec Discount 336864
Li Bi Rec Promo 423111
Li Bi Rec Shared Cost 29898
Li Bi Term Total Direct Cost 29890
Li Bi Term Total Net Price 2724
Li Bi Total Shared Cost 29898
Li Dd Rec Net Price 2726
```

In [32]:
```
subset_close_final[numerical_data].describe() #Get a summary of numberical data for closed opportunity data after removing ou
```

Out[32]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Discount	Li Bi Rec Promo	Li Bi Rec Shared Cost	Li Bi T Total D
count	423111.000000	4.231110e+05	423111.000000	423111.000000	423111.000000	423111.000000	423111.000000	423111.0	423111.000000	423111.00
mean	32.453671	2.318461e+04	57.499974	537.289101	363.097227	1374.254602	-99.467399	0.0	311.285397	2200.09
std	11.782536	9.764191e+04	82.716609	694.821755	505.483972	1964.688859	303.057782	0.0	371.576078	2829.70
min	2.000000	0.000000e+00	0.000000	0.000000	-577.020000	150.000000	-2970.000000	0.0	0.000000	0.00
25%	36.000000	0.000000e+00	27.160000	133.000000	119.000000	450.000000	0.000000	0.0	115.930000	1124.86
50%	36.000000	1.000000e+04	31.500000	285.000000	191.880000	805.000000	0.000000	0.0	138.880000	1520.85
75%	36.000000	1.000000e+04	49.510000	695.000000	390.000000	1431.000000	0.000000	0.0	361.130000	2075.97
max	132.000000	1.200000e+06	1013.090000	7000.000000	6035.480000	22807.000000	184.005000	0.0	3141.470000	33165.94

Since Li Bi Rec Promo varaible is all 0s after outlier removing, we decide to remove this variable. In addition, Li Bi Rec Discount has majority of 0s which is highly skewed with only influentical points, it will be removed as well.

Concat both Open_Opportunity and Closed_Opportunity after removing outliers from Closed_Opportunity data set, named subset_all_final

In [33]:
```
subset_all_final = pd.concat([subset_close_final,subset_open])
#Drop varaible Li Bi Rec Promo since all 0 values
subset_all_final = subset_all_final.drop(['Li Bi Rec Promo','Li Bi Rec Discount'], axis=1)
print(subset_all_final.shape)
```

```
(759115, 45)
```

In [34]:
```
#Redefine numerical_data column after droppping variable Li Bi Rec Promo
numerical_data  = ['Term',
                   'Commitment Amount',
                   'Li Bi Rec Direct Cost',
                   'Li Bi Rec List Price',
                   'Li Bi Rec Net Price',
                   'Price',
                   'Li Bi Rec Shared Cost',
                   'Li Bi Term Total Direct Cost',
                   'Li Bi Term Total Net Price',
                   'Li Bi Total Shared Cost',
                   'Li Dd Rec Net Price']
```

For all data (combined closed and open opportunity data), we redefine four types of dataframe after moving the outliers from closed opportunity data

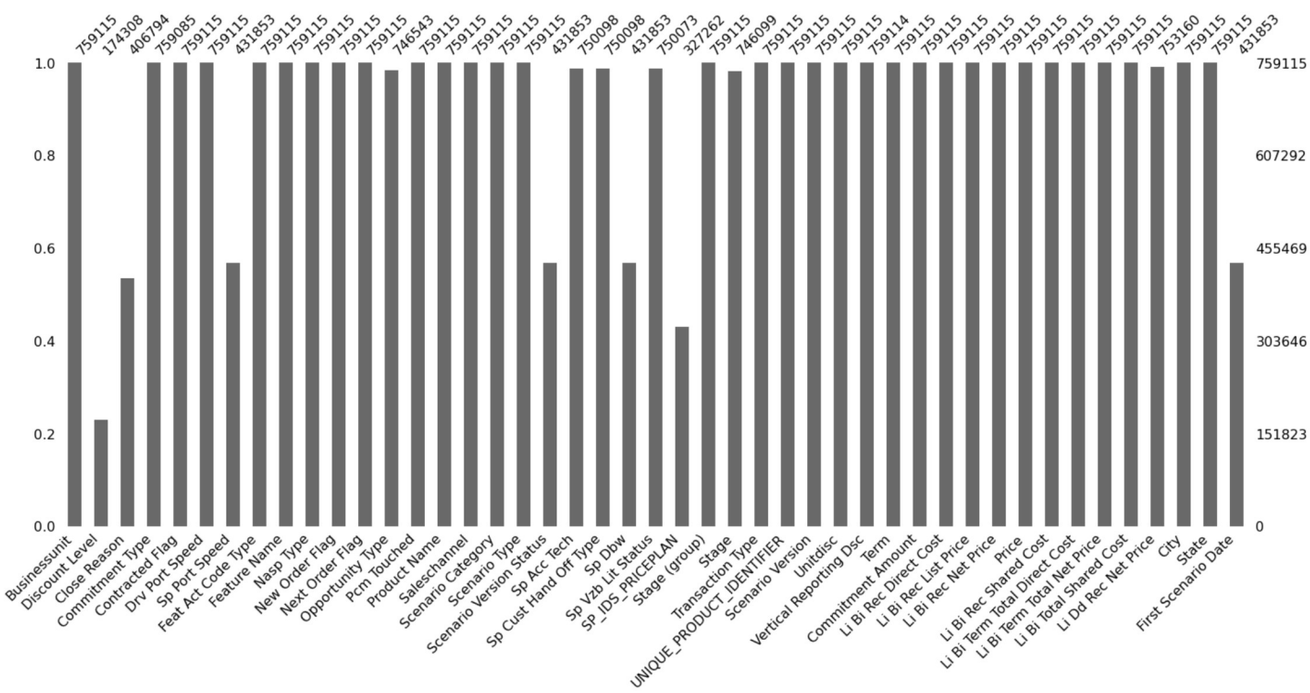
In [35]:
```
categorical_df_all_final = subset_all_final[categorical_data]
numerical_df_all_final = subset_all_final[numerical_data]
geo_df_all_final = subset_all_final[geo_data]
temporal_df_all_final = subset_all_final[temporal_data]
print(categorical_df_all_final.shape,numerical_df_all_final.shape,geo_df_all_final.shape,temporal_df_all_final.shape)
#Output the data shape for closed_opportunity_final after removing the outliers
```

```
(759115, 31) (759115, 11) (759115, 2) (759115, 1)
```

Missing Value Imputation on all data (both Open_Opportunity and Closed_Opportunity)

In [36]:
```
#plot to show the missing data
msno.bar(subset_all_final)
```

Out[36]:
```
<AxesSubplot:>
```



In [37]:
```
all_data_na = subset_all_final.isnull().sum()
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Missing Value Count' :all_data_na})
missing_data #Give us a list of missing value count
```

Out[37]:

	Missing Value Count
Discount Level	584807
SP_IDS_PRICEPLAN	431853
Close Reason	352321
Sp Port Speed	327262
Scenario Version Status	327262
Sp Dbw	327262
First Scenario Date	327262
Stage	13016
Opportunity Type	12572
Sp Vzb Lit Status	9042
Sp Acc Tech	9017
Sp Cust Hand Off Type	9017
Li Dd Rec Net Price	5955
Commitment Type	30
Vertical Reporting Dsc	1

In [38]:
```
all_data_na_ratio = (subset_all_final.isnull().sum() / len(subset)) * 100
all_data_na_ratio = all_data_na_ratio.drop(all_data_na_ratio[all_data_na_ratio == 0].index).sort_values(ascending=False)
missing_data_ratio = pd.DataFrame({'Missing Ratio' :all_data_na_ratio})
missing_data_ratio #Give us a list of missing value ratio
```

Out[38]:

	Missing Ratio
Discount Level	75.362795
SP_IDS_PRICEPLAN	55.651949
Close Reason	45.402834
Sp Port Speed	42.173536
Scenario Version Status	42.173536
Sp Dbw	42.173536
First Scenario Date	42.173536
Stage	1.677343
Opportunity Type	1.620126
Sp Vzb Lit Status	1.165223
Sp Acc Tech	1.162001
Sp Cust Hand Off Type	1.162001
Li Dd Rec Net Price	0.767408
Commitment Type	0.003866
Vertical Reporting Dsc	0.000129

In [39]:
```
#need to imputate numeric_data with median for numerical_df_all_final
numerical_df_all_final.fillna(numerical_df_all_final.median(), inplace=True)
```

In [40]:
```
#need to imputate categorical_data with Unknown for categorical_df_all_final
categorical_df_all_final = categorical_df_all_final.fillna("Unknown")
```

In [41]:
```
#no need to imputate geo_df_all_final
geo_df_all_final.isnull().sum()
```

Out[41]:
```
City     0
State    0
dtype: int64
```

In [42]:
```
#need to imputate temporal data with "01/01/1900" for temporal_df_all_final
temporal_df_all_final = temporal_df_all_final.fillna("01/01/1900")
```

For all data (combined closed and open opportunity data), we concatenate all four types data dataframe after the imputation, named subset_all_final_com

In [43]:
```
#Concat all four types of variable
subset_all_final_com = pd.concat([numerical_df_all_final,categorical_df_all_final,geo_df_all_final,temporal_df_all_final], ax
subset_all_final_com.shape
```

Out[43]:
```
(759115, 45)
```

In [44]:
```
all_data_na2 = subset_all_final_com.isnull().sum()
all_data_na2 = all_data_na2.drop(all_data_na2[all_data_na2 == 0].index).sort_values(ascending=False)
missing_data2 = pd.DataFrame({'Missing Value Count' :all_data_na2})
missing_data2 #check the null value again, there is no missing value now
```

Out[44]:

	Missing Value Count

There is no missing value after the imputation.

Resplit open and close opportunity data after imputing the missing value

In [45]:
```
#define Open_Opportunity_final for the open opportunity and Close_Opportunity_final for the closed opportunity.
Open_Opportunity_final = subset_all_final_com.loc[subset_all_final_com['Stage (group)'] != '5 Closed Disqualified, 5 Closed L
Close_Opportunity_final = subset_all_final_com.loc[subset_all_final_com['Stage (group)'] == '5 Closed Disqualified, 5 Closed
print('Open_Opportunity_final: ', Open_Opportunity_final.shape)
print('Close_Opportunity_final: ', Close_Opportunity_final.shape)
```

```
Open_Opportunity_final:  (336004, 45)
Close_Opportunity_final:  (423111, 45)
```

Now we can see we have 336004 records for open opportunity and 423111 for closed 423111 opportunity, both has 45 columns.

In [46]:
```
#All variables for Open_Opportunity only, we redefine four types of data after moving the outliers and missing value imputati
categorical_df_open_final = Open_Opportunity_final[categorical_data]
numerical_df_open_final = Open_Opportunity_final[numerical_data]
geo_df_open_final = Open_Opportunity_final[geo_data]
temporal_df_open_final = Open_Opportunity_final[temporal_data]
print(categorical_df_open_final.shape,numerical_df_open_final.shape,geo_df_open_final.shape,temporal_df_open_final.shape)
```

```
(336004, 31) (336004, 11) (336004, 2) (336004, 1)
```

Above is the processed Open_Opportunity data we will use for the pricing prediction, data shapes displayed above.

In [47]:
```
#All variables for Closed_Opportunity only, we redefine four types of data after moving the outliers and missing value imputa
categorical_df_close_final = Close_Opportunity_final[categorical_data]
numerical_df_close_final = Close_Opportunity_final[numerical_data]
geo_df_close_final = Close_Opportunity_final[geo_data]
temporal_df_close_final = Close_Opportunity_final[temporal_data]
print(categorical_df_close_final.shape,numerical_df_close_final.shape,geo_df_close_final.shape,temporal_df_close_final.shape)
```

```
(423111, 31) (423111, 11) (423111, 2) (423111, 1)
```

Above is the processed Closed_Opportunity data we will use for the futuer modeling, data shapes displayed above.

Initial Skewness Analysis

In [48]:
```
# Find skewed numerical features
skew_features = numerical_df_close_final.apply(lambda x: skew(x)).sort_values(ascending=False)

high_skew = skew_features[skew_features > 0.5]
skew_index = high_skew.index

print("There are {} numerical features with Skew > 0.5 :".format(high_skew.shape[0]))
skewness = pd.DataFrame({'Skew' :high_skew})
skew_features
```

Out[48]:
```
There are 10 numerical features with Skew > 0.5 :
Commitment Amount               8.115800
Price                           5.300470
Li Bi Term Total Net Price      4.590650
Li Dd Rec Net Price             4.405546
Li Bi Rec Net Price             4.379341
Li Bi Rec Direct Cost           4.028222
Li Bi Term Total Direct Cost    4.009419
Li Bi Rec List Price            3.169740
Li Bi Total Shared Cost         2.468189
Li Bi Rec Shared Cost           2.229652
Term                           -0.051668
dtype: float64
```

In [49]:
```
#Analysis for original target variable
fig,(ax1,ax2,ax3) = plt.subplots(1,3,constrained_layout=True,figsize=(10,3))

y1 = numerical_df_close_final.Price
sns.boxplot(y = y1, ax=ax1)
ax1.set_title("Boxplot of Price")
sns.distplot(y1, ax=ax2,kde=False, fit=stats.norm)
ax2.set_title("Histrogram of Price")
stats.probplot(y1,plot=ax3)
ax3.set_title("QQplot of Price")
print("Kurtosis: %f" % numerical_df_close_final.Price.kurt())
print("Skewness: %f" % numerical_df_close_final.Price.skew())
```

```
Kurtosis: 39.324452
Skewness: 5.300489
```

Feature Engineering on both open and closed opportunity data

Log Transformation

StandardScaler

Apply log transformation

In [50]:
```
#define the original numberical columns including target variable Price
original_num_columns = numerical_df_close_final.columns

#define the original numberical columns excluding target variable Price
original_num_columns1 = numerical_df_close_final.drop(['Price'], axis=1).columns
```

In [51]:
```
#converting each numberical columns into its log variation for open opportunity
for column in original_num_columns1:
        numerical_df_open_final[column + '_log'] = np.log1p(numerical_df_open_final[column])
log_columns = numerical_df_open_final.columns[numerical_df_open_final.columns.str.contains('_log')].tolist()
# getting a quick pick at the skewed data
numerical_df_open_final.head(2)
```

Out[51]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	...	Term_log	Commitment Amount_log	Li Bi Rec Direct Cost_log	Li Bi Rec List Price_log
0	60.0	0	1631.46	4712.0	2450.240	6000.0	2532.99	98198.85	137213.440	151979.40	...	4.110874	0.0	7.397843	8.458080
1	36.0	0	194.68	2428.0	1073.176	2428.0	1167.44	7008.48	36487.984	42027.84	...	3.610918	0.0	5.276481	7.795235

2 rows × 21 columns

In [52]:
```
#converting each numberical into its log variation for closed opportunity
for column in original_num_columns1:
        numerical_df_close_final[column + '_log'] = np.log1p(numerical_df_close_final[column])
log_columns = numerical_df_close_final.columns[numerical_df_close_final.columns.str.contains('_log')].tolist()
# getting a quick pick at the skewed data
numerical_df_close_final.head(2)
```

Out[52]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	...	Term_log	Commitment Amount_log	Li Bi Rec Direct Cost_log	Li Bi Rec List Price_log	Li Bi Rec Net Price_log	Li
0	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	...	3.610918	0.0	3.620601	5.72815	5.72815	5.
1	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	...	3.610918	0.0	3.620601	5.72815	5.72815	5.

2 rows × 21 columns

After applying the log transformation, there might be some small possibilities na value occurs, check the null values again.

In [53]:
```
numerical_df_open_final.isnull().sum()[numerical_df_open_final.isnull().sum()>0]
```

Out[53]:
```
Li Bi Rec Net Price_log           2
Li Bi Term Total Net Price_log    2
dtype: int64
```

In [54]:
```
numerical_df_close_final.isnull().sum()[numerical_df_close_final.isnull().sum()>0]
```

Out[54]:
```
Li Bi Rec Net Price_log           6
Li Bi Term Total Net Price_log    6
dtype: int64
```

In [55]:
```
#Inpute nan value with 0 after log trnasformation
numerical_df_open_final = numerical_df_open_final.fillna(0)
numerical_df_close_final = numerical_df_close_final.fillna(0)
```

Apply StandardScaler for both closed and open opportunity based on log transformed data

In [56]:
```
# Standardizing open opportunity data
scaler1 = StandardScaler()
numerical_scaled = scaler1.fit_transform(numerical_df_open_final[log_columns].values)
new_col = [l + "_ss" for l in log_columns]
numerical_df_open_final_scaled = pd.DataFrame(numerical_scaled, columns=new_col)
#add scaled data to our numerical_df_open_final data
numerical_df_open_final = pd.concat([numerical_df_open_final, numerical_df_open_final_scaled], axis=1)
pd.set_option("display.max_columns", numerical_df_open_final.shape[1])
# Log transformation on target variable
numerical_df_open_final['Price_log'] = np.log1p(numerical_df_open_final.Price)
numerical_df_open_final.head(2)
```

Out[56]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	Li Dd Rec Net Price	Term_log	Commitment Amount_log	Li Bi Rec Direct Cost_log	Li
0	60.0	0	1631.46	4712.0	2450.240	6000.0	2532.99	98198.85	137213.440	151979.40	2450.240	4.110874	0.0	7.397843	8.4
1	36.0	0	194.68	2428.0	1073.176	2428.0	1167.44	7008.48	36487.984	42027.84	1073.176	3.610918	0.0	5.276481	7.7

2 rows × 32 columns

In [57]:
```
# Standardizing closed opportunity data
scaler2 = StandardScaler()
numerical_scaled = scaler2.fit_transform(numerical_df_close_final[log_columns].values)
new_col = [l + "_ss" for l in log_columns]
numerical_df_close_final_scaled = pd.DataFrame(numerical_scaled, columns=new_col)
#add scaled data to our numerical_df_close_final data
numerical_df_close_final = pd.concat([numerical_df_close_final, numerical_df_close_final_scaled], axis=1)
pd.set_option("display.max_columns", numerical_df_close_final.shape[1])
# Log transformation on target variable
numerical_df_close_final['Price_log'] = np.log1p(numerical_df_close_final.Price)
numerical_df_close_final.head(2)
```

Out[57]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	Li Dd Rec Net Price	Term_log	Commitment Amount_log	Li Bi Rec Direct Cost_log	Li Bi Rec List Price_log	...	Li
0	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	306.4	3.610918	0.0	3.620601	5.72815	...	7.2412
1	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	306.4	3.610918	0.0	3.620601	5.72815	...	7.2412

2 rows × 32 columns

In [58]:
```
#only for original open data columns
numerical_df_open_final[original_num_columns].head(2)
```

Out[58]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	Li Dd Rec Net Price
0	60.0	0	1631.46	4712.0	2450.240	6000.0	2532.99	98198.85	137213.440	151979.40	2450.240
1	36.0	0	194.68	2428.0	1073.176	2428.0	1167.44	7008.48	36487.984	42027.84	1073.176

In [59]:
```
#only for original closed data columns
numerical_df_close_final[original_num_columns].head(2)
```

Out[59]:

	Term	Commitment Amount	Li Bi Rec Direct Cost	Li Bi Rec List Price	Li Bi Rec Net Price	Price	Li Bi Rec Shared Cost	Li Bi Term Total Direct Cost	Li Bi Term Total Net Price	Li Bi Total Shared Cost	Li Dd Rec Net Price
0	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	306.4
1	36.0	0	36.36	306.4	306.4	766.0	199.58	1394.86	11030.4	7184.88	306.4

In [60]:
```
#Create a shortlist for the modeling named new_col
new_col.append('Price_log')
new_col
```

Out[60]:
```
['Term_log_ss',
 'Commitment Amount_log_ss',
 'Li Bi Rec Direct Cost_log_ss',
 'Li Bi Rec List Price_log_ss',
 'Li Bi Rec Net Price_log_ss',
 'Li Bi Rec Shared Cost_log_ss',
 'Li Bi Term Total Direct Cost_log_ss',
 'Li Bi Term Total Net Price_log_ss',
 'Li Bi Total Shared Cost_log_ss',
 'Li Dd Rec Net Price_log_ss',
 'Price_log']
```

In [61]:
```
#only for scaled open data columns
numerical_df_open_final[new_col].head(2)
```

Out[61]:

	Term_log_ss	Commitment Amount_log_ss	Li Bi Rec Direct Cost_log_ss	Li Bi Rec List Price_log_ss	Li Bi Rec Net Price_log_ss	Li Bi Rec Shared Cost_log_ss	Li Bi Term Total Direct Cost_log_ss	Li Bi Term Total Net Price_log_ss	Li Bi Total Shared Cost_log_ss	Li Dd Rec Net Price_log_ss	Price_log
0	1.357507	-1.354157	3.021751	2.380703	2.074098	1.66757	2.210963	2.404383	1.510823	2.085517	8.699681
1	0.201407	-1.354157	1.300576	1.742471	1.259573	1.20408	0.872121	1.206298	0.970956	1.264629	7.795235

In [62]:
```
#only for scaled closed data columns
numerical_df_close_final[new_col].head(2)
```

Out[62]:

	Term_log_ss	Commitment Amount_log_ss	Li Bi Rec Direct Cost_log_ss	Li Bi Rec List Price_log_ss	Li Bi Rec Net Price_log_ss	Li Bi Rec Shared Cost_log_ss	Li Bi Term Total Direct Cost_log_ss	Li Bi Term Total Net Price_log_ss	Li Bi Total Shared Cost_log_ss	Li Dd Rec Net Price_log_ss	Price_log
0	0.415467	-1.417895	0.07978	0.007497	0.329546	0.203624	0.153701	0.460816	0.315839	0.333021	6.642487
1	0.415467	-1.417895	0.07978	0.007497	0.329546	0.203624	0.153701	0.460816	0.315839	0.333021	6.642487

Below workdata_open is the unscaled data for the final OPEN opportunity data

In [63]:
```
workdata_open = pd.concat([categorical_df_open_final,numerical_df_open_final[original_num_columns],geo_df_open_final,temporal
workdata_open.shape
```

Out[63]:
```
(336004, 45)
```

Below workdata is the unscaled data for the final CLOSED opportunity data we will be used for further EDA

In [64]:
```
#this is the unscaled data for closed opportunity
workdata = pd.concat([categorical_df_close_final,numerical_df_close_final[original_num_columns],geo_df_close_final,temporal_d
workdata.shape
```

Out[64]:
```
(423111, 45)
```

Below workdata_open_scaled is the scaled data for the final OPEN opportunity data we will be used for the prediction

In [65]:
```
#this is the logscaled open opportunity data which we will be using for the predicting
workdata_open_scaled = pd.concat([categorical_df_open_final,numerical_df_open_final[new_col],geo_df_open_final,temporal_df_op
workdata_open_scaled.shape
```

Out[65]:
```
(336004, 45)
```

Below workdata_scaled is the scaled data for the final CLOSED opportunity data we will be used for the modeling

In [66]:
```
#this is the logscaled closed data which we will be using in the modeling
workdata_scaled = pd.concat([categorical_df_close_final,numerical_df_close_final[new_col],geo_df_close_final,temporal_df_clos
workdata_scaled.shape
```

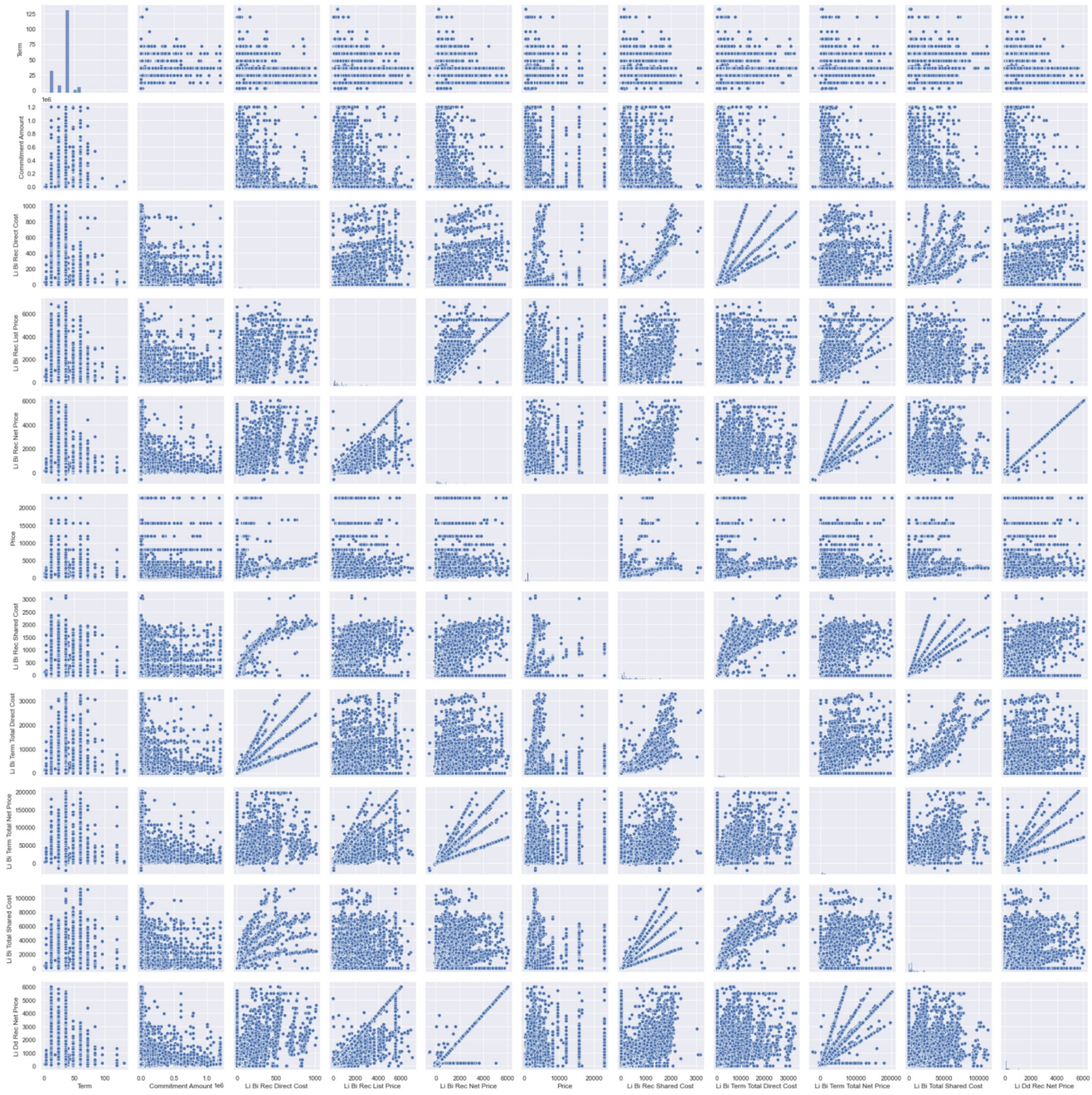
Out[66]:
```
(423111, 45)
```

In [67]:
```
workdata_scaled.info() #check the workdata_scaled info again before modeling
```

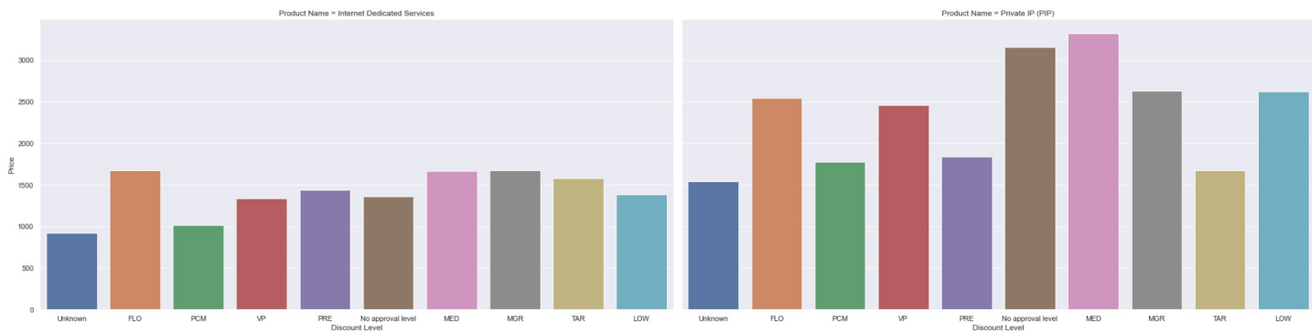
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 423111 entries, 0 to 423110
Data columns (total 45 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Businessunit                        423111 non-null  object
 1   Discount Level                      423111 non-null  object
 2   Close Reason                        423111 non-null  object
 3   Commitment Type                     423111 non-null  object
 4   Contracted Flag                     423111 non-null  object
 5   Drv Port Speed                      423111 non-null  object
 6   Sp Port Speed                       423111 non-null  object
 7   Feat Act Code Type                  423111 non-null  object
 8   Feature Name                        423111 non-null  object
 9   Nasp Type                           423111 non-null  object
 10  New Order Flag                      423111 non-null  object
 11  Next Order Flag                     423111 non-null  object
 12  Opportunity Type                    423111 non-null  object
 13  Pcm Touched                         423111 non-null  object
 14  Product Name                        423111 non-null  object
 15  Saleschannel                        423111 non-null  object
 16  Scenario Category                   423111 non-null  object
 17  Scenario Type                       423111 non-null  object
 18  Scenario Version Status             423111 non-null  object
 19  Sp Acc Tech                         423111 non-null  object
 20  Sp Cust Hand Off Type               423111 non-null  object
 21  Sp Dbw                              423111 non-null  object
 22  Sp Vzb Lit Status                   423111 non-null  object
 23  SP_IDS_PRICEPLAN                    423111 non-null  object
 24  Stage (group)                       423111 non-null  object
 25  Stage                               423111 non-null  object
 26  Transaction Type                    423111 non-null  object
 27  UNIQUE_PRODUCT_IDENTIFIER           423111 non-null  object
 28  Scenario Version                    423111 non-null  object
 29  Unitdisc                            423111 non-null  object
 30  Vertical Reporting Dsc              423111 non-null  object
 31  Term_log_ss                         423111 non-null  float64
 32  Commitment Amount_log_ss            423111 non-null  float64
 33  Li Bi Rec Direct Cost_log_ss        423111 non-null  float64
 34  Li Bi Rec List Price_log_ss         423111 non-null  float64
 35  Li Bi Rec Net Price_log_ss          423111 non-null  float64
 36  Li Bi Rec Shared Cost_log_ss        423111 non-null  float64
 37  Li Bi Term Total Direct Cost_log_ss 423111 non-null  float64
 38  Li Bi Term Total Net Price_log_ss   423111 non-null  float64
 39  Li Bi Total Shared Cost_log_ss      423111 non-null  float64
 40  Li Dd Rec Net Price_log_ss          423111 non-null  float64
 41  Price_log                           423111 non-null  float64
 42  City                                423111 non-null  object
 43  State                               423111 non-null  object
 44  First Scenario Date                 423111 non-null  object
dtypes: float64(11), object(34)
memory usage: 145.3+ MB
```

EDA on original closed(training) dataset - workdata

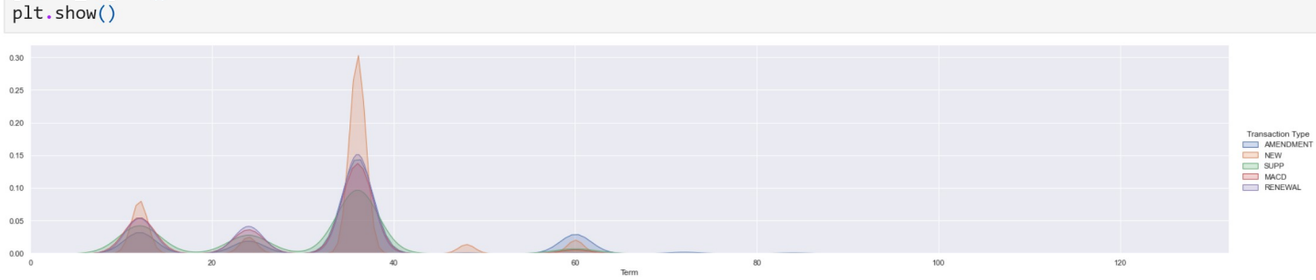
In [68]:
```
#scatterplot to magnify and look at the selected variable
sns.set()
sns.pairplot(workdata[original_num_columns])
plt.show()
```



In [69]:
```
#EDA with respect to target variable for workdataset
g = sns.factorplot(x="Discount Level", y="Price", col="Product Name",
                   data=workdata,kind="bar", ci=None, size=7, aspect=2)
plt.show()
```



In [70]:
```
g1 = sns.FacetGrid(workdata, hue = 'Transaction Type', aspect=5,size=5 )
g1.map(sns.kdeplot, 'Term', shade= True )
g1.set(xlim=(0 , workdata['Term'].max()))
g1.add_legend()
plt.show()
```



In [71]:
```
pd.crosstab(workdata['Discount Level'],workdata['Stage (group)']).style.background_gradient(cmap='PuBu')
```

Out[71]:

Stage (group)	5 Closed Disqualified, 5 Closed Lost, 5 Closed Won
Discount Level	
FLO	8843
LOW	278
MED	46
MGR	1736
No approval level	1668
PCM	47991
PRE	188
TAR	636
Unknown	336864
VP	24861

In [72]:
```
# crosstab to get some summaries from subset2
pd.crosstab(workdata['Discount Level'],[workdata['Product Name'],workdata['Transaction Type']]).style.background_gradient(cma
```

Out[72]:

Product Name	Internet Dedicated Services					Private IP (PIP)				
Transaction Type	**AMENDMENT**	**MACD**	**NEW**	**RENEWAL**	**SUPP**	**AMENDMENT**	**MACD**	**NEW**	**RENEWAL**	**SUPP**
Discount Level										
FLO	76	26	7166	141	1	16	0	1262	151	4
LOW	0	7	199	18	0	0	0	40	14	0
MED	0	6	32	2	0	0	0	6	0	0
MGR	18	110	370	48	1	147	35	915	90	2
No approval level	32	1	644	0	0	25	0	966	0	0
PCM	2257	248	18110	148	114	3528	1768	19874	1150	794
PRE	4	6	146	8	0	0	0	12	12	0
TAR	0	2	424	8	0	0	0	197	5	0
Unknown	9444	1880	109174	1303	235	16061	8149	182536	5490	2592
VP	621	614	14609	500	18	864	286	6729	573	47

In [73]:
```
pd.crosstab(workdata['Drv Port Speed'],workdata['Scenario Type']).style.background_gradient(cmap='PuBu')
```

Out[73]:

Scenario Type	ILLUSTRATIVE	NEXTORDER	PROPOSED	QUOTE_TO_ORDER	TRANSACTIONAL
Drv Port Speed					
1 Gbps	7165	449	6087	1040	5907
1 Mbps	1512	272	2566	1334	848
1.5 Gbps	16	0	16	10	2
1.5 Mbps	326	662	398	216	560
10 Gbps	211	43	154	50	86
10 Mbps	81235	9801	25088	18254	4418
100 Mbps	18606	1293	15911	6836	8669
15 Mbps	264	112	382	250	38
150 Mbps	426	71	375	191	125
155 Mbps	4	1	9	0	0
2 Gbps	145	56	129	65	117
2 Mbps	11262	952	4095	2481	563
2.5 Gbps	1	0	1	11	0
20 Mbps	20409	1973	11898	8067	2443
200 Mbps	3667	615	2076	2897	1156
256 Kbps	8	10	4	10	44
3 Gbps	19	12	4	0	0
3 Mbps	3606	935	1756	2580	839
30 Mbps	11500	378	6359	573	377
300 Mbps	832	182	721	286	522
4 Gbps	9	2	1	0	2
4 Mbps	6795	269	223	1888	76
4.6 Mbps	4	2	10	20	13
40 Mbps	557	213	791	431	150
400 Mbps	185	41	179	92	72
45 Mbps	15	2	21	23	5
5 Gbps	7	1	2	0	0
5 Mbps	4868	1165	5070	1835	998
50 Mbps	25493	2115	13223	5010	4879
500 Mbps	3178	288	1097	306	1000
512 Kbps	0	0	0	0	2
6 Mbps	1922	784	311	529	93
60 Mbps	1143	8	217	105	15
600 Mbps	425	14	252	32	29
64 Kbps	0	182	0	0	12
7 Gbps	0	1	0	0	0
7 Mbps	78	4	40	26	6
70 Mbps	132	16	240	141	29
700 Mbps	27	9	7	4	5
8 Mbps	140	68	207	74	37
80 Mbps	1926	429	378	1172	50
800 Mbps	49	8	36	31	13
9 Mbps	8	2	8	2	2
90 Mbps	44	2	26	4	4

In [74]:
```
pd.crosstab(workdata['Sp Port Speed'],workdata['Scenario Version Status']).style.background_gradient(cmap='PuBu')
```

Out[74]:

Scenario Version Status / Sp Port Speed	PCM_ACCEPTED	PCM_APPROVED	PCM_DENIED	PCM_VALIDATED	PRICEBOOK_CREATED	PRICE_OPTIONS_FAILED	PRICE_OPTIONS_RETURN
1 Gbps	34	249	9	89	1538	0	
1 Mbps	4	532	0	22	1314	0	1
1.5 Gbps	0	2	0	1	6	0	
1.5 Mbps	156	150	18	16	838	0	
10 Gbps	0	3	0	0	8	0	
10 Mbps	840	6022	104	1437	29110	6	97
100 Mbps	1428	1495	138	502	7537	0	2
15 Mbps	2	120	2	158	334	0	
150 Mbps	22	82	0	22	334	0	
155 Mbps	1	1	0	0	5	0	
2 Gbps	2	12	0	0	91	0	
2 Mbps	220	1152	434	720	3174	0	4
2.5 Gbps	0	0	0	0	3	0	
20 Mbps	584	1047	14	405	8319	4	17
200 Mbps	80	320	14	78	1797	0	1
256 Kbps	0	4	0	4	36	0	
3 Gbps	0	0	0	0	14	0	
3 Mbps	254	638	136	308	2562	0	
30 Mbps	60	757	16	4405	1187	0	11
300 Mbps	34	117	4	24	637	0	
4 Gbps	0	0	0	0	0	0	
4 Mbps	50	240	8	784	1174	0	
4.6 Mbps	6	0	0	0	26	0	
40 Mbps	12	99	0	202	708	0	
400 Mbps	15	17	2	20	122	0	
45 Mbps	4	0	0	4	34	0	
5 Gbps	0	0	0	0	0	0	
5 Mbps	130	1260	520	635	2286	2	
50 Mbps	504	880	46	462	8277	0	2
500 Mbps	29	93	1	28	386	0	
512 Kbps	0	0	0	0	0	0	
6 Mbps	36	80	6	96	1270	0	
60 Mbps	8	26	0	22	260	0	2
600 Mbps	1	14	1	2	63	0	
64 Kbps	0	0	0	0	178	0	
7 Mbps	0	12	0	2	94	0	
70 Mbps	4	18	0	16	192	0	
8 Mbps	12	0	0	94	168	0	
80 Mbps	14	28	4	62	534	0	
800 Mbps	0	12	0	1	28	0	
9 Mbps	0	8	0	2	4	0	
90 Mbps	2	8	0	4	34	0	
Unknown	0	0	0	0	0	0	

In [75]:
```
pd.crosstab(workdata['Nasp Type'],[workdata['Transaction Type'],workdata['Unitdisc']], rownames=['Nasp Type']).style.backgrou
```

Out[75]:

Transaction Type	AMENDMENT						MACD															
Unitdisc	FLO	List	PCM	PRE	Sales	VP	FLO	LOW	List	MED	PCM	PRE	Sales	TAR	VP	FLO	LOW	List	MED	PCM	PRE	Sa
Nasp Type																						
C1 - US	0	137	615	0	19	111	0	0	5	0	51	0	3	0	16	60	0	1638	0	10290	1	
C1 - US MID MARKETS	17	2	28	0	0	91	0	0	6	0	2	0	2	0	71	392	2	279	1	296	10	
C1 CORP - ASIA	0	0	0	0	0	0	0	0	4	0	12	0	0	0	0	0	0	3	0	47	0	
C1 CORP - CANADA	0	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	208	0	0	0	
C1 CORP - EMEA	0	14	97	0	0	8	0	0	11	0	23	0	0	0	2	0	0	165	0	439	0	
C1 CORP - LATAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	6	0	31	0	
C1 CORP GOV - EMEA	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	
C1 EMEA MED BUS	0	0	6	0	0	0	0	0	6	0	6	0	0	0	0	0	0	7	0	19	0	
C1 MM PARTNER - EMEA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	
C1 VWAG2 - ASIA	0	0	3	0	0	4	0	0	0	0	0	0	0	0	0	1	0	2	0	16	0	
C1 VWAG2 - EMEA	0	2	57	0	0	0	0	0	1	0	6	0	0	0	0	0	0	41	0	8423	0	
C1 VWAG2 - LATAM	0	0	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	530	0	
C1 VWAG2 - US	0	0	3343	0	0	0	0	0	2	0	2	0	0	0	0	0	0	1	0	2968	0	
C2 - US	0	80	1768	0	17	188	0	0	21	0	165	0	7	0	66	76	1	5636	0	13621	3	
C2 - US MID MARKETS	62	57	285	4	16	505	5	0	27	0	53	0	26	0	211	3458	42	1772	11	2517	60	1
C2 CORP - ASIA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	13	0	
C2 CORP - EMEA	0	0	31	0	0	0	0	0	0	0	6	0	0	0	0	0	0	60	0	126	0	
C2 MM PARTNER - EMEA	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	
C2 VWAG2 - ASIA	0	0	31	0	0	2	0	0	0	0	0	0	2	0	0	0	0	7	0	22	0	
E1 - US	0	15	1063	0	0	54	0	0	2	0	1608	0	2	0	40	14	0	9628	0	19895	0	
E1 CORP - EMEA	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	
E1 VWAG2 - ASIA	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	1	0	
E1 VWAG2 - ASIA CS	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	
E1 VWAG2 - EMEA	0	0	170	0	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	189	0	
E1 VWAG2 - US	0	45	4337	0	0	0	0	0	0	0	1651	0	0	0	2	0	0	9308	0	51943	0	
E1 VWAG2 - US CS	0	0	2824	0	0	0	0	0	0	0	76	0	0	0	0	0	0	15	0	3761	0	
E2 - US	0	8	1450	0	11	17	0	0	1	0	1101	0	1	0	3	58	0	1749	0	18364	0	
E2 CORP - EMEA	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	20	0	
E2 VWAG2 - ASIA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
E2 VWAG2 - ASIA CS	0	0	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	4	0	3545	0	
E2 VWAG2 - EMEA	0	4	293	0	0	0	0	0	3	0	118	0	0	0	0	0	0	80	0	2063	0	
E2 VWAG2 - EMEA CS	0	0	181	0	0	0	0	0	2	0	20	0	0	0	0	0	0	3	0	431	0	
E2 VWAG2 - US	0	1	617	0	0	0	0	0	2	0	380	0	0	0	0	1	0	1889	0	24420	0	
E2 VWAG2 - US CS	0	0	329	0	0	0	0	0	0	0	97	0	0	0	0	0	0	1	0	742	0	
E3 - US	0	44	4277	0	1	70	0	0	6	0	2532	0	1	0	4	9	0	9076	0	29196	0	
E3 - US CS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	1927	0	
E3 CORP - ASIA	0	0	10	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	32	0	
E3 CORP - EMEA	0	0	94	0	0	0	0	0	2	0	10	0	2	0	0	0	0	0	0	219	0	
E3 VWAG2 - ASIA	0	0	41	0	0	2	0	0	2	0	314	0	0	0	4	0	0	10	0	351	0	
E3 VWAG2 - ASIA CS	0	0	4	0	0	0	0	0	0	0	5	0	0	0	0	0	0	2	0	233	0	
E3 VWAG2 - CANADA	0	0	14	0	0	0	0	0	0	0	8	0	0	0	0	0	0	1	0	84	0	
E3 VWAG2 - EMEA	0	0	293	0	0	4	0	0	1	0	19	0	2	0	0	2	0	80	0	476	0	
E3 VWAG2 - US	0	20	951	0	0	2	0	0	0	0	738	0	0	0	3	0	0	1702	0	19653	0	
E4 - US	0	106	4998	0	69	234	0	0	5	0	1309	0	1	0	15	25	0	3466	0	16932	0	
E4 - US CS	0	10	191	0	0	0	0	0	0	0	1	0	0	0	0	0	0	10	0	417	0	
E4 CORP - ASIA	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	3	0	50	0	
E4 CORP - CANADA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	
E4 CORP - EMEA	0	2	22	0	0	0	0	0	4	0	9	0	0	0	0	0	0	11	0	130	0	

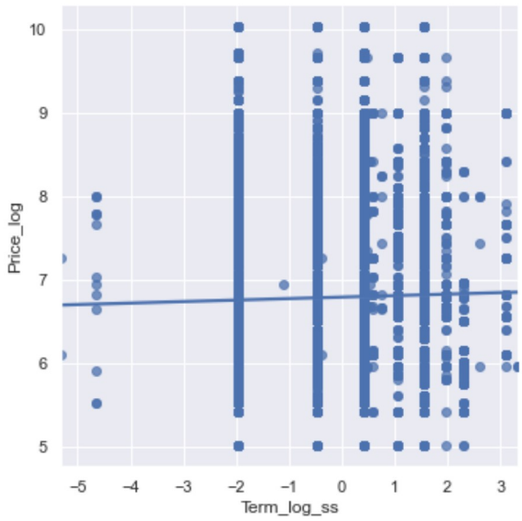
Transaction Type	AMENDMENT													MACD									
Unitdisc	FLO	List	PCM	PRE	Sales	VP	FLO	LOW	List	MED	PCM	PRE	Sales	TAR	VP	FLO	LOW	List	MED	PCM	PRE	Sa
Nasp Type																						
E4 VWAG2 - ASIA	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	4	0	59	0	
E4 VWAG2 - ASIA CS	0	4	11	0	0	0	0	0	1	0	3	0	0	0	0	0	0	109	0	806	0	
E4 VWAG2 - CANADA	0	0	688	0	4	0	0	0	0	0	10	0	0	0	0	0	0	1	0	804	0	
E4 VWAG2 - EMEA	0	0	41	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	45	0	
E4 VWAG2 - EMEA CS	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	2	0	10	0	
E4 VWAG2 - US	0	0	500	0	0	0	0	0	0	0	1050	0	0	0	0	0	0	411	0	2160	0	
E4 VWAG2 - US CS	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	
ENTERPRISE (SELL WITH)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	47	0	
F1 ARMY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	712	0	26	0	1
F1 COURTS	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	
F1 DHS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	860	0	48	0	
F1 DISA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	
F1 HHS	0	0	0	0	0	0	0	0	0	0	44	0	0	0	0	0	0	197	0	217	0	
F1 INTERIOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	0	19	0	
F1 JUSTICE	0	0	0	0	0	0	0	0	1	0	6	0	0	0	0	0	0	318	0	6	0	
F1 NAVY	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	769	0	0	0	
F1 SELECT ACCOUNTS	0	0	14	0	12	16	0	0	0	0	0	0	2	0	0	0	0	70	0	16	0	
F1 TREASURY	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	7	0	
F1 USPS	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
F1 VA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	5	0	
F2 AIR FORCE	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	265	0	3	0	
F2 DEFENSE AGENCIES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	
F2 EMEA	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	
F2 EMEA SELECT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F2 ENERGY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	0	0	0	
F2 FAA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F2 GSA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	
F2 INTERNATIONAL AGENCIES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	
F2 INTERNATIONAL BANKS	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	
F2 LABOR	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	18	0	4	0	
F2 LEGISLATIVE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	869	0	6	0	
F2 NUCLEAR REG COMM	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	34	0	
F2 OSD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	
F2 PENTAGON	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F2 SSA	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	8	0	207	0	
F2 STATE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	
F3 BANKS	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F3 COMMERCE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	59	0	
F3 COMMODITY FUTURES TRAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	16	0	
F3 CORP FOR NATL SERVICE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	
F3 EDUCATION	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	
F3 HUD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	
F3 NASA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F3 SMALL AGENCIES 1	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	8	0	18	0	
F3 SMALL AGENCIES 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	7	0	
F3 SMALL AGENCIES 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
F3 SMITHSONIAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	6	0	

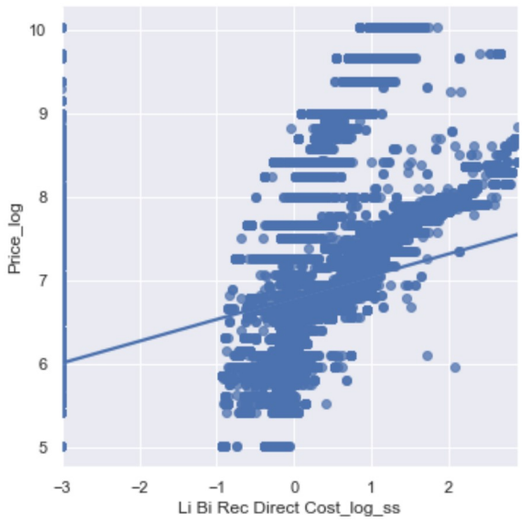
Transaction Type	AMENDMENT						MACD															
Unitdisc	FLO	List	PCM	PRE	Sales	VP	FLO	LOW	List	MED	PCM	PRE	Sales	TAR	VP	FLO	LOW	List	MED	PCM	PRE	Sa
Nasp Type																						
F3 TRIBES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GE1 COMMONWEALTH OF MA	0	0	8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	16	0	
GE1 COMMONWEALTH OF PA	0	0	325	0	1	0	0	0	0	0	0	0	0	0	0	0	0	250	0	25	0	
GE1 COMMONWEALTH OF VA	0	0	160	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	867	0	
GE1 NEW YORK CITY	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	
GE1 NYC PUBLIC SAFETY	0	0	33	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	30	0	
GE1 STATE OF CA	0	0	4	0	0	0	0	0	0	0	17	0	0	0	0	0	0	8	0	29	0	
GE1 STATE OF MARYLAND	0	1	36	0	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	24	0	
GE1 STATE OF NEW JERSEY	0	6	0	0	4	0	0	0	0	0	0	0	0	0	0	14	0	7	0	17	0	
GE1 STATE OF NEW YORK	0	0	40	0	2	2	0	0	0	0	9	0	0	0	2	0	0	16	0	781	0	
GE2 BALTIMORE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	
GE2 BOSTON	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GE2 CENTRAL FLORIDA	0	0	6	0	0	2	0	0	0	0	4	0	0	0	2	0	0	0	0	4	0	
GE2 FAIRFAX COUNTY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GE2 LOS ANGELES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	2	0	
GE2 PHILADELPHIA	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	
GE2 SLED CORE ACCOUNT	0	0	14	0	4	0	0	0	0	0	0	0	4	0	0	0	0	4	0	163	0	
GE2 ST OF WEST VIRGINIA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GE2 STATE OF DELAWARE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	
GE2 STATE OF FLORIDA	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	149	0	2	0	
GE3 911	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	48	0	
GE3 SE NORTH TERRITORY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	252	0	33718	0	
GE3 SLED CORE ACCOUNT	0	0	22	0	2	18	0	0	0	0	4	0	2	0	0	1	0	23	0	181	0	
GE3 SOUTH FLORIDA	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	
GE4 ED HIGHER - INSIDE	0	0	0	0	0	3	0	0	1	0	0	0	0	0	3	0	0	18	0	0	0	
GE4 ED LOWER - INSIDE	0	0	0	0	0	30	0	0	2	0	1	0	0	0	16	0	0	29	0	1	0	
GE4 EDU NONBILLER	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	12	0	2	0	
GE4 MA CARAHSOFT - INSIDE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
GE4 MUNI NONBILLER	0	0	0	0	0	20	0	0	1	0	0	0	0	0	3	7	0	51	0	44	0	
GE4 MUNICIPAL - INSIDE	0	0	0	0	0	6	0	0	0	0	0	0	0	0	2	0	0	3	0	0	0	
GE4 NON ED KINGCOM - INSI	0	0	0	0	0	0	0	0	0	0	0	0	7	0	4	2	0	35	0	0	0	
GE4 SLED CORE ACCOUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	
INDIRECT - MB	1	12	4	0	0	30	4	0	31	0	6	1	3	0	80	506	9	120	1	17	7	
INDIRECT - MID MARKETS	0	0	0	0	0	22	0	0	1	0	0	0	0	0	9	69	2	15	0	5	4	
INDIRECT NEW LOGO - MB	2	0	0	0	0	1	0	0	2	0	0	0	0	0	17	371	3	37	0	64	3	
INDIRECT NEW LOGO - MM	0	7	1	0	0	4	0	0	0	0	0	0	4	0	34	54	0	6	0	0	1	
M1 - CANADA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	
M1 MM - ASIA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

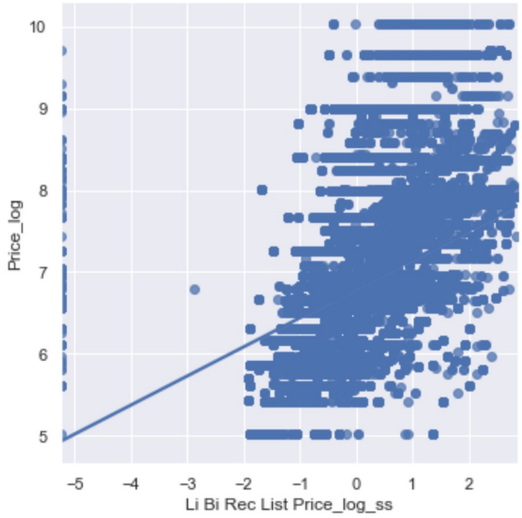
Transaction Type	AMENDMENT						MACD															
Unitdisc	FLO	List	PCM	PRE	Sales	VP	FLO	LOW	List	MED	PCM	PRE	Sales	TAR	VP	FLO	LOW	List	MED	PCM	PRE	Sa
Nasp Type																						
MASS MARKETS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MB EMEA MED BUS	0	0	25	0	0	0	0	0	2	0	8	0	0	0	5	0	0	135	0	78	0	
MEDIUM BUSINESS	10	3	23	0	0	16	17	7	197	6	14	5	74	2	283	3072	175	768	25	681	62	1
MID MARKETS	0	0	0	0	0	4	0	0	0	0	0	0	0	0	1	3	1	1	0	0	0	
PARTNER - SLED	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72	0	
SFDC GENERIC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	2	7	0	56	0	
SFDC GENERIC VPP - MB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
SLED (SELL WITH)	0	0	251	0	0	0	0	0	0	0	0	0	0	0	0	0	0	288	0	665	0	
VERIZON PARTNER PROGRAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	12	0	0	5	
VWA - LATAM	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	13	0	
VZ PARTNER PROGRAM - EMEA	0	0	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	
XO - ENTERPRISE	0	0	1	0	0	0	0	0	0	0	1	0	0	0	2	0	0	5	0	62	0	
XO - LATAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	
XO - MEDIUM BUSINESS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	11	0	0	2	
XO - MID MARKETS	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	14	1	4	0	32	0	
XO - SLED	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
XO INDIRECT - MEDIUM BUSI	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	5	1	9	0	0	0	
XO INDIRECT - MID MARKETS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	129	0	4	0	0	0	

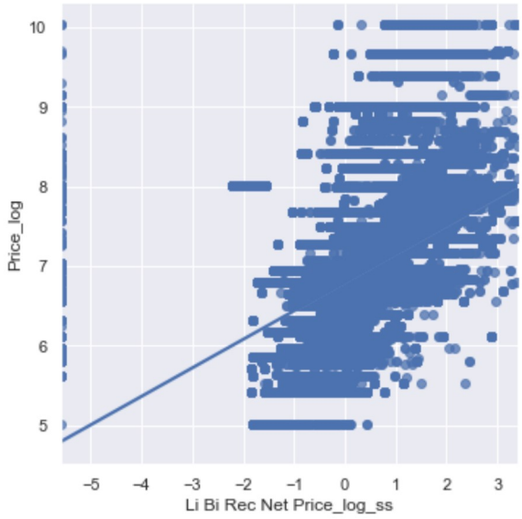
Skewness analysis for colsed_opportunity data after standard scaling use workdata_scaled dataframe

In [76]:
```
#Looking at the data after cleaning the outliers, applying the standard scaling and logtransformation
chart1 = sns.lmplot(x='Term_log_ss', y='Price_log', data=workdata_scaled, scatter_kws={'alpha':0.75})
chart2 = sns.lmplot(x='Li Bi Rec Direct Cost_log_ss', y='Price_log', data=workdata_scaled, scatter_kws={'alpha':0.75})
chart3 = sns.lmplot(x='Li Bi Rec List Price_log_ss', y='Price_log', data=workdata_scaled, scatter_kws={'alpha':0.75})
chart4 = sns.lmplot(x='Li Bi Rec Net Price_log_ss', y='Price_log', data=workdata_scaled, scatter_kws={'alpha':0.75})
```







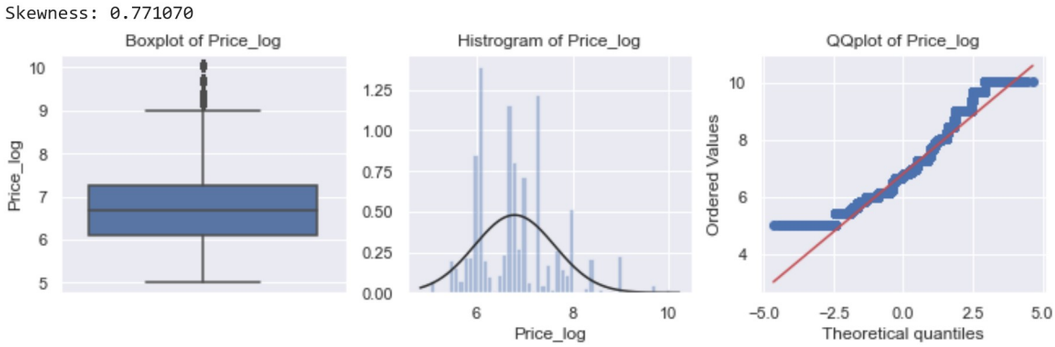


In [77]:
```
# Find skewed numerical features
skew_features = workdata_scaled[new_col].apply(lambda x: skew(x)).sort_values(ascending=False)

high_skew = skew_features[skew_features > 0.5]
skew_index = high_skew.index

print("There are {} numerical features with Skew > 0.5 :".format(high_skew.shape[0]))
skewness = pd.DataFrame({'Skew' :high_skew})
skew_features
```

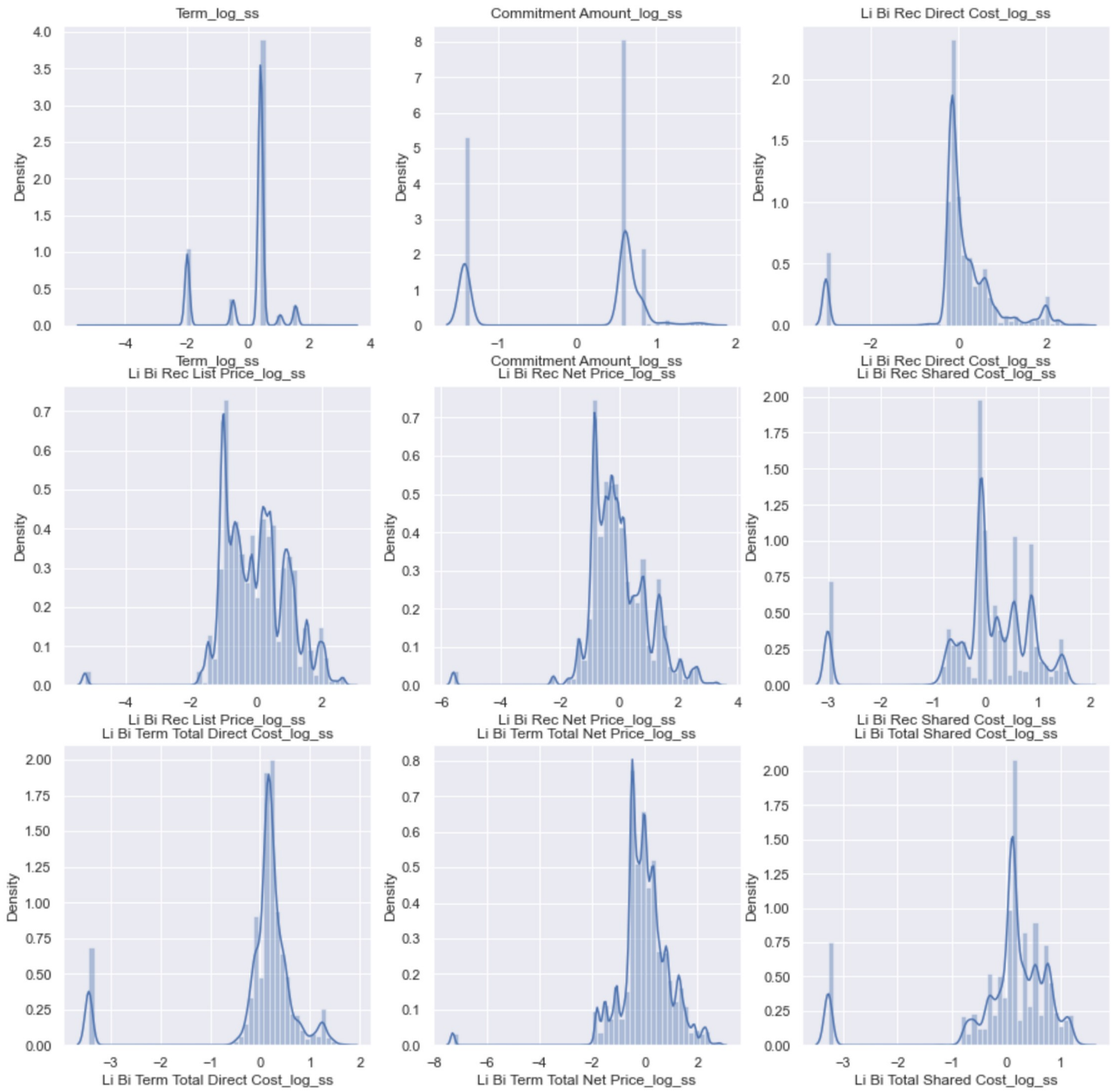
Out[77]:
```
There are 1 numerical features with Skew > 0.5 :
Price_log                            0.771067
Li Bi Rec List Price_log_ss         -0.427744
Li Bi Rec Net Price_log_ss          -0.490673
Li Dd Rec Net Price_log_ss          -0.491569
Commitment Amount_log_ss            -0.653563
Term_log_ss                         -1.115770
Li Bi Rec Direct Cost_log_ss        -1.381361
Li Bi Rec Shared Cost_log_ss        -1.646710
Li Bi Term Total Net Price_log_ss   -2.246836
Li Bi Total Shared Cost_log_ss      -2.339472
Li Bi Term Total Direct Cost_log_ss -2.715584
dtype: float64
```

In [78]:
```
#Analysis again for original target variable after the feature engineering
fig,(ax1,ax2,ax3) = plt.subplots(1,3,constrained_layout=True,figsize=(10,3))

y1 = workdata_scaled.Price_log
sns.boxplot(y = y1, ax=ax1)
ax1.set_title("Boxplot of Price_log")
sns.distplot(y1, ax=ax2,kde=False, fit=stats.norm)
ax2.set_title("Histrogram of Price_log")
stats.probplot(y1,plot=ax3)
ax3.set_title("QQplot of Price_log")
print("Kurtosis: %f" % workdata_scaled.Price_log.kurt())
print("Skewness: %f" % workdata_scaled.Price_log.skew())
```

```
Kurtosis: 0.719859
Skewness: 0.771070
```



In [79]:
```
# Looking at the standard scaled version of our variables
fig, axes = plt.subplots(nrows = 3, ncols = 3)
axes = axes.flatten()
fig.set_size_inches(15, 15)

for ax, col in zip(axes, new_col):
    sns.distplot(workdata_scaled[col], ax = ax)
    ax.set_title(col)
```



After perfroming the data imputation, log transformation, and standard scaler and skewness analysis on the numberical data, workdata_scaled is the one we will use for modeling, workdata_open_scaled is the one we will use for prediction. Look at the distribution plots, skewness and statistical plots, we can see our dataset workdata_scaled is in good shape for the modeling.

In [80]:
```
workdata_scaled.describe()
```

Out[80]:

	Term_log_ss	Commitment Amount_log_ss	Li Bi Rec Direct Cost_log_ss	Li Bi Rec List Price_log_ss	Li Bi Rec Net Price_log_ss	Li Bi Rec Shared Cost_log_ss	Li Bi Term Total Direct Cost_log_ss	Li Bi Term Total Net Price_log_ss	Li Bi Total Shared Cost_log_ss	Li
count	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.231110e+05	4.
mean	-4.036543e-13	7.519515e-14	4.706136e-14	-1.813165e-14	-5.972888e-14	1.316271e-14	-1.399463e-14	-4.907823e-14	9.465577e-15	-3
std	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.
min	-5.317725e+00	-1.417895e+00	-3.019670e+00	-5.244050e+00	-5.587247e+00	-3.010028e+00	-3.433163e+00	-7.279626e+00	-3.277697e+00	-5.
25%	4.154675e-01	-1.417895e+00	-1.622267e-01	-7.537285e-01	-6.420902e-01	-1.235102e-01	4.722150e-02	-4.618556e-01	-9.676810e-02	-6
50%	4.154675e-01	6.090137e-01	-3.952117e-02	-5.865755e-02	-1.518846e-01	-1.487154e-02	1.965058e-01	-1.925444e-02	1.261321e-01	-1
75%	4.154675e-01	6.090137e-01	3.379422e-01	7.567024e-01	5.780263e-01	5.617671e-01	3.505472e-01	4.758755e-01	5.468384e-01	5
max	3.335186e+00	1.662558e+00	2.905758e+00	2.873089e+00	3.405028e+00	1.871643e+00	1.722948e+00	2.880613e+00	1.431187e+00	3.

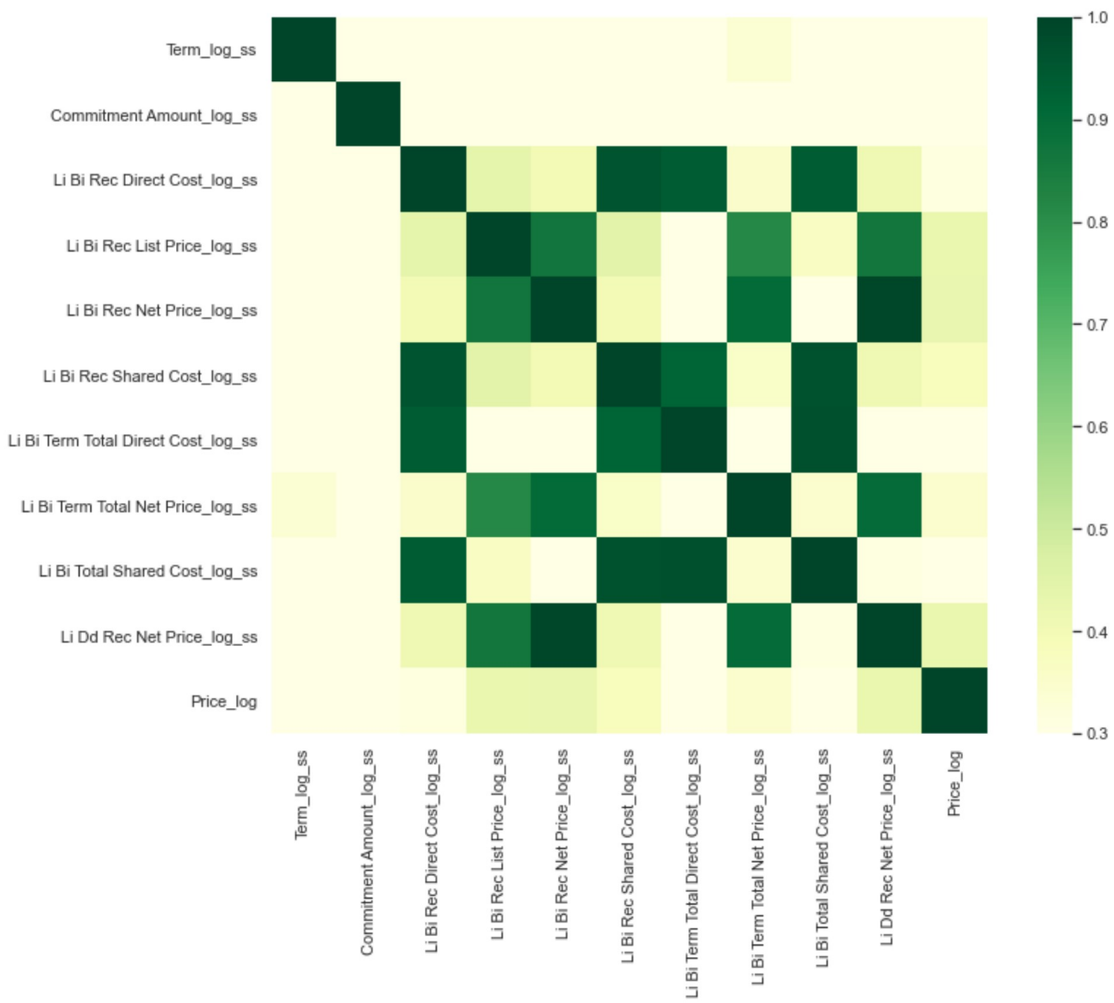
In [81]:
```
workdata_open_scaled.describe()
```

Out[81]:

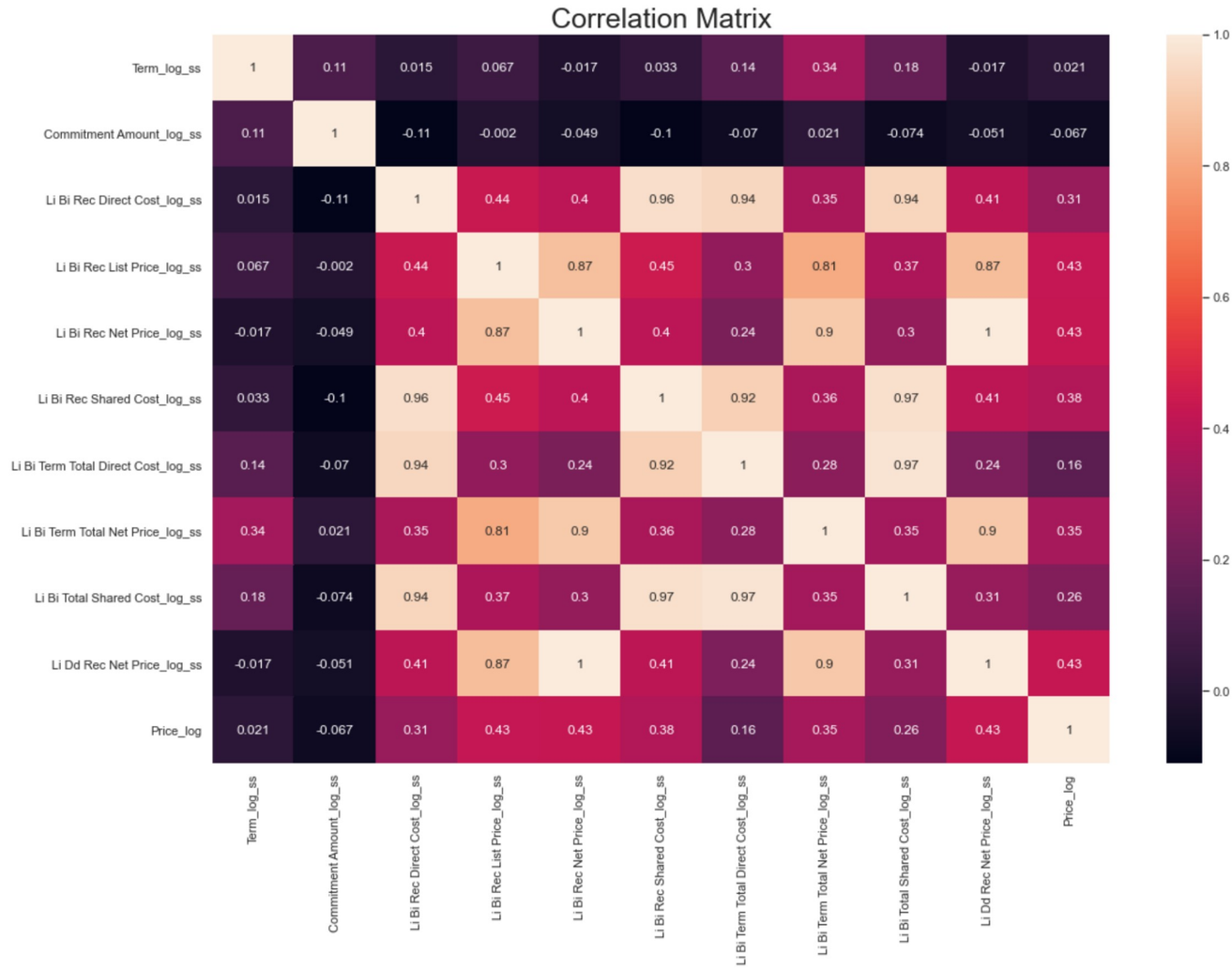
	Term_log_ss	Commitment Amount_log_ss	Li Bi Rec Direct Cost_log_ss	Li Bi Rec List Price_log_ss	Li Bi Rec Net Price_log_ss	Li Bi Rec Shared Cost_log_ss	Li Bi Term Total Direct Cost_log_ss	Li Bi Term Total Net Price_log_ss	Li Bi Total Shared Cost_log_ss	Li
count	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.360040e+05	3.
mean	-2.327295e-13	1.495777e-13	-8.944446e-15	-4.636478e-14	-7.260692e-14	1.691578e-14	-2.934315e-14	3.259291e-14	1.793997e-14	9
std	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.
min	-4.942821e+00	-1.354157e+00	-2.980515e+00	-5.763310e+00	-5.630789e+00	-3.024980e+00	-3.619029e+00	-8.295604e+00	-3.500418e+00	-5.
25%	2.014070e-01	-1.354157e+00	-2.772090e-01	-6.960348e-01	-7.521252e-01	-4.891274e-01	-1.418157e-02	-6.902513e-01	-1.473504e-01	-7
50%	2.014070e-01	5.752688e-01	-1.268216e-01	-4.624733e-02	-1.757550e-01	-1.061611e-01	1.387084e-01	-3.830301e-02	4.571469e-02	-1
75%	2.014070e-01	5.752688e-01	4.158511e-01	7.659523e-01	5.838829e-01	7.952632e-01	3.479046e-01	5.797458e-01	6.514675e-01	5
max	2.941311e+00	2.841801e+00	3.855351e+00	5.045128e+00	5.451408e+00	2.258374e+00	2.697504e+00	5.067969e+00	1.898472e+00	5.

Checking correlation matrix using heatmap to find features influencing the target variable "Price" the most.

In [82]:
```
sns.heatmap(workdata_scaled.corr(),square=True, vmax = 1, vmin = 0.3, xticklabels=True, yticklabels=True, cmap='YlGn')
fig=plt.gcf()
fig.set_size_inches(12,9)
plt.show()
```



In [83]:
```
plt.figure(figsize=(18,12))
sns.heatmap((workdata_scaled.corr()),annot=True)
plt.title('Correlation Matrix', fontsize = 25)
plt.show()
```



Use OneHot encoder for categorical variables for CLOSED opportunity, define workdata_scaled_digit and we will use workdata_scaled_digit dataframe for modeling.

In [84]:
```
OneHotencoder = OneHotEncoder()
OneHotencoder = ce.OneHotEncoder(cols=categorical_data,\
handle_unknown='ignore',return_df=True,use_cat_names=True)
workdata_scaled_digit = OneHotencoder.fit_transform(workdata_scaled)
```

Use OneHot encoder for categorical variables for OPEN opportunity, define testdata_scaled_digit and we will use testdata_scaled_digit dataframe for prediction.

In [85]:
```
OneHotencoder = OneHotEncoder()
OneHotencoder = ce.OneHotEncoder(cols=categorical_data,\
handle_unknown='ignore',return_df=True,use_cat_names=True)
testdata_scaled_digit = OneHotencoder.fit_transform(workdata_open_scaled)
```

Using Label encodering for geo_data and temporal_data that may contain information in ordering set, applied to workdata_scaled_digit dataset (CLOSED opportunity).

In [86]:
```
from sklearn.preprocessing import LabelEncoder

cols = ['City','State','First Scenario Date']

for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(workdata_scaled_digit[c].values))
    workdata_scaled_digit[c] = lbl.transform(list(workdata_scaled_digit[c].values))

# shape
print('Shape all_closed_data: {}'.format(workdata_scaled_digit.shape))
```

```
Shape all_closed_data: (423111, 524)
```

Using Label encodering for geo_data and temporal_data that may contain information in ordering set, applied to testdata_scaled_digit (OPEN opportunity).

In [87]:
```
for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(testdata_scaled_digit[c].values))
    testdata_scaled_digit[c] = lbl.transform(list(testdata_scaled_digit[c].values))

# shape
print('Shape all_open_data: {}'.format(testdata_scaled_digit.shape))
```

```
Shape all_open_data: (336004, 514)
```

Check and remove the dummy variables that either not in workdata_scaled_digit or not in testdata_scaled_digit

In [88]:
```
# Get missing columns, exist in workdata_scaled_digit but not in testdata_scaled_digit dataset
missing_cols1 = set( workdata_scaled_digit.columns ) - set( testdata_scaled_digit.columns )
missing_cols1 = missing_cols1- set(workdata_scaled.columns)
```

In [89]:
```
# Get missing columns, exist in testdata_scaled_digit but not in workdata_scaled_digit dataset
missing_cols2 = set( testdata_scaled_digit.columns ) - set( workdata_scaled_digit.columns )
missing_cols2 = missing_cols2 - set(workdata.columns)
```

In [90]:
```
# We exlucde missing_cols1 and missing_cols2 from workdata_scaled_digit and testdata_scaled_digit dataset
Col_to_exlcude = missing_cols1.union(missing_cols2)
list(Col_to_exlcude)
```

Out[90]:
```
['Nasp Type_F3 HUD',
 'Nasp Type_E3 - US CS',
 'Nasp Type_F2 INTERNATIONAL AGENCIES',
 'Drv Port Speed_6 Gbps',
 'Close Reason_Renewal: Retained',
 'Nasp Type_GE2 DIST OF COLUMBIA (DC)',
 'Close Reason_Lost: Recycle Campaign Oppty',
 'Nasp Type_C1 VWAG2 - LATAM',
 'Feat Act Code Type_NEW',
 'Stage_3 Propose',
 'Nasp Type_F3 EPA',
 'Nasp Type_F2 STATE',
 'Close Reason_No Bid',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-7 Gbps-Ethernet',
 'Stage_5 Closed Disqualified',
 'Opportunity Type_Unknown',
 'Sp Port Speed_9 Gbps',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-100 Mbps-TDM',
 'Close Reason_Lost: Competitor Capabilities',
 'Nasp Type_F3 SMALL AGENCIES 4',
 'Nasp Type_MASS MARKETS',
 'Nasp Type_C1 MM PARTNER - EMEA',
 'Nasp Type_F2 DEFENSE AGENCIES',
 'Commitment Type_TRC',
 'Sp Port Speed_622 Mbps',
 'Stage_5 Closed Won',
 'Nasp Type_F2 GSA',
 'Nasp Type_GE2 STATE OF TEXAS',
 'Scenario Version_21',
 'Nasp Type_XO - SLED',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-6 Gbps-Ethernet',
 'Stage (group)_5 Closed Disqualified, 5 Closed Lost, 5 Closed Won',
 'Stage_4 Negotiate',
 'Stage_0 Identify',
 'Stage_1 Qualify',
 'Sp Vzb Lit Status_Third Party Datacenter POP',
 'Scenario Version_25',
 'Nasp Type_F3 CORP FOR NATL SERVICE',
 'Sp Port Speed_6 Gbps',
 'UNIQUE_PRODUCT_IDENTIFIER_Internet Dedicated Services-6 Gbps-Ethernet',
 'Nasp Type_F1 TREASURY',
 'Nasp Type_ENTERPRISE (SELL WITH)',
 'Sp Port Speed_155 Mbps',
 'Stage (group)_Null, 0 Identify, 1 Qualify and 3 more',
 'Drv Port Speed_8 Gbps',
 'Drv Port Speed_9 Gbps',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-9 Gbps-Ethernet',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-622 Mbps-TDM',
 'Nasp Type_F2 TRANSPORTATION',
 'Nasp Type_F2 OSD',
 'Stage_5 Closed Lost',
 'Stage_Unknown',
 'Close Reason_Renewal: Disconnected',
 'Nasp Type_E1 VWAG2 - ASIA',
 'Nasp Type_F3 SECS AND EXCHG COMM',
 'Close Reason_Disqualified: Oppty Withdrawal',
 'Nasp Type_F3 TRIBES',
 'Scenario Version_18',
 'Close Reason_Sold: Account Relationship',
 'Nasp Type_C2 CORP - CANADA',
 'Nasp Type_GE2 FAIRFAX COUNTY',
 'Nasp Type_F3 SMITHSONIAN',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-155 Mbps-TDM',
 'Drv Port Speed_622 Mbps',
 'Stage_2 Solve',
 'Sp Vzb Lit Status_AU-1F Collo',
 'Nasp Type_F1 VA',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-8 Gbps-Ethernet',
 'Close Reason_Won: VEC-eComm Order',
 'Vertical Reporting Dsc_Unknown',
 'Commitment Type_Unknown',
 'Vertical Reporting Dsc_RETAIL/SPECIALTY SOFT: APPAREL STORES',
 'Close Reason_DQ: TUI Invalid',
 'Nasp Type_F3 COMMODITY FUTURES TRAD',
 'Vertical Reporting Dsc_HOSPITALITY',
 'UNIQUE_PRODUCT_IDENTIFIER_Internet Dedicated Services-8 Gbps-Ethernet',
 'Nasp Type_GE2 LOS ANGELES',
 'Nasp Type_SFDC GENERIC - SLED',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-50 Mbps-TDM',
 'Scenario Version_43',
 'Discount Level_ECM',
 'Sp Port Speed_7 Gbps',
 'UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-500 Mbps-TDM',
 'Sp Port Speed_8 Gbps',
 'Nasp Type_F3 NASA',
 'Nasp Type_XO - LATAM']
```

In [91]:
```
#Remove variable in testdata_scaled_digit but not in workdata_scaled_digit
cols1 = [col for col in workdata_scaled_digit.columns if col not in Col_to_exlcude]
workdata_scaled_digit = workdata_scaled_digit[cols1]
print('Shape all_closed_data: {}'.format(workdata_scaled_digit.shape))
```

```
Shape all_closed_data: (423111, 476)
```

In [92]:
```
#Remove variable in workdata_scaled_digit but not in testdata_scaled_digit
cols2 = [col for col in testdata_scaled_digit.columns if col not in Col_to_exlcude]
testdata_scaled_digit = testdata_scaled_digit[cols2]
print('Shape all_open_data: {}'.format(testdata_scaled_digit.shape))
```

```
Shape all_open_data: (336004, 476)
```

Data Split into train and validation

In [93]:
```
#Define predictive dataset with dropping the predictive variable
Pred_data = testdata_scaled_digit.drop(['Price_log'],axis=1)
Pred_data.shape
```

Out[93]:
```
(336004, 475)
```

In [94]:
```
#Split the train and test data from workdata_scaled_digit dataset
y = workdata_scaled_digit.Price_log
X = workdata_scaled_digit.drop(['Price_log'],axis=1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, \
                                    test_size = 0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[94]:
```
((338488, 475), (84623, 475), (338488,), (84623,))
```

In [95]:
```
#Create five folds for the validation
from sklearn.model_selection import GridSearchCV, \
train_test_split, KFold

kfold =KFold(n_splits=5, shuffle=True, random_state=42)
cnt = 1
# split()  method generate indices to split data into training and test set.
for train_index, test_index in kfold.split(X, y):
    print(f'Fold:{cnt}, Train set: {len(train_index)}\
    , Test set:{len(test_index)}')
    cnt += 1
```

```
Fold:1, Train set: 338488    , Test set:84623
Fold:2, Train set: 338489    , Test set:84622
Fold:3, Train set: 338489    , Test set:84622
Fold:4, Train set: 338489    , Test set:84622
Fold:5, Train set: 338489    , Test set:84622
```

Regression

In [96]:
```
# Importing sklearn libraries
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, Lasso, LassoCV, LassoLarsCV, LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler

#from sklearn.linear_model import ElasticNet, Lasso, Ridge, Lasso regressions as well as metrics
from sklearn.ensemble import RandomForestRegressor,  GradientBoostingRegressor
from sklearn.kernel_ridge import KernelRidge
from sklearn.base import BaseEstimator, TransformerMixin, RegressorMixin, clone
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import mean_squared_error

from yellowbrick.regressor import ResidualsPlot
from yellowbrick.model_selection import LearningCurve
from yellowbrick.regressor import PredictionError
```

In [97]:
```
#Define helper functions that used in the modeling
def cross_val(model):
    cv_rmse = np.sqrt(-cross_val_score(model, X, y, \
    scoring="neg_mean_squared_error", cv=kfold))
    model_score = model.score(X,y)
    return cv_rmse.mean(), cv_rmse.std(), model_score
def evaluate_model(true, predicted):
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    R2_squared = metrics.r2_score(true, predicted)
    return mse,rmse, R2_squared
def print_evaluate_results(true, predicted):
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    R2_squared = metrics.r2_score(true, predicted)
    print('MSE:', round(mse,5), 'RMSE:', round(rmse,5), \
          'R2_Squared', round(R2_squared,5))
def print_cross_val(model):
    cv_rmse = np.sqrt(-cross_val_score(model, X, y, \
    scoring="neg_mean_squared_error", cv=kfold))
    model_score = model.score(X,y)
    print('CV Results:', 'Mean:',round(cv_rmse.mean(),5),\
'Std:',round(cv_rmse.std(),5), 'Model R2_Squared', round(model_score,5))

def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val = \
    train_test_split(X, y, test_size=0.33, random_state=10)
    train_errors, val_errors = [], []
    for m in range(1, len(X_train)):
        model.fit(X_train[:m], y_train[:m])
        y_train_predict = model.predict(X_train[:m])
        y_val_predict = model.predict(X_val)
        train_errors.append(mean_squared_error\
                            (y_train[:m], y_train_predict))
        val_errors.append(mean_squared_error(y_val, y_val_predict))
    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="train")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")
    plt.legend(loc="upper right", fontsize=14)
    plt.xlabel("Training set size", fontsize=14)
    plt.ylabel("RMSE", fontsize=14)
```

Linear Regression

In [98]:
```
#Fit linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train)
print(lin_reg.intercept_)
```

```
7.151543192961742
```

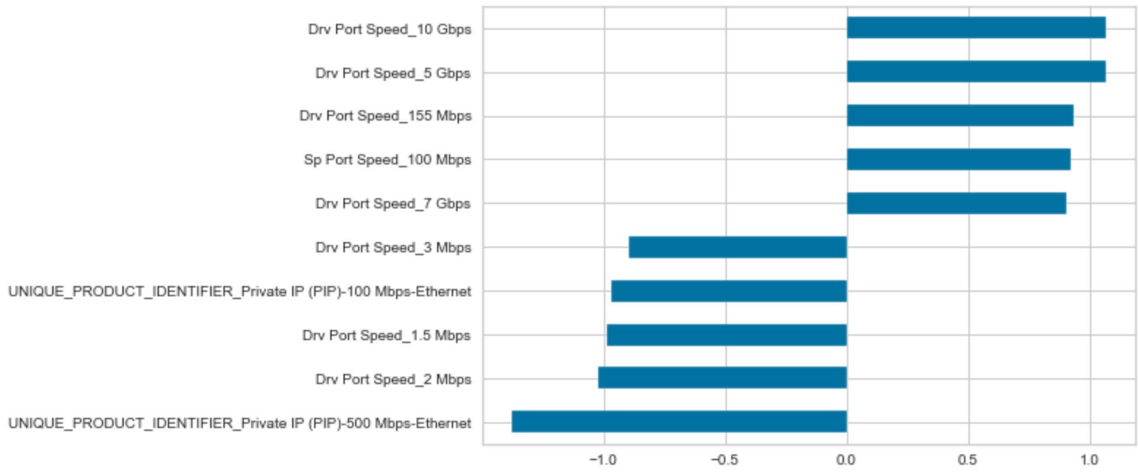
In [99]:
```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df.sort_values(['Coefficient'], ascending=False)
```

Out[99]:

	Coefficient
Drv Port Speed_10 Gbps	1.065717
Drv Port Speed_5 Gbps	1.065334
Drv Port Speed_155 Mbps	0.934221
Sp Port Speed_100 Mbps	0.921773
Drv Port Speed_7 Gbps	0.902984
...	...
Drv Port Speed_3 Mbps	-0.891894
UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-100 Mbps-Ethernet	-0.968390
Drv Port Speed_1.5 Mbps	-0.982924
Drv Port Speed_2 Mbps	-1.018306
UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-500 Mbps-Ethernet	-1.377023

475 rows × 1 columns

In [100…
```
en_coefs = pd.Series(lin_reg.coef_, index = X_train.columns)
imp_coefs = pd.concat([en_coefs.sort_values().head(),\
(en_coefs[en_coefs==0]),en_coefs.sort_values().tail()])
imp_coefs.plot(kind = "barh")
plt.show()
```



In [101…
```
y_pred_train=lin_reg.predict(X_train)
y_pred_test=lin_reg.predict(X_test)
```

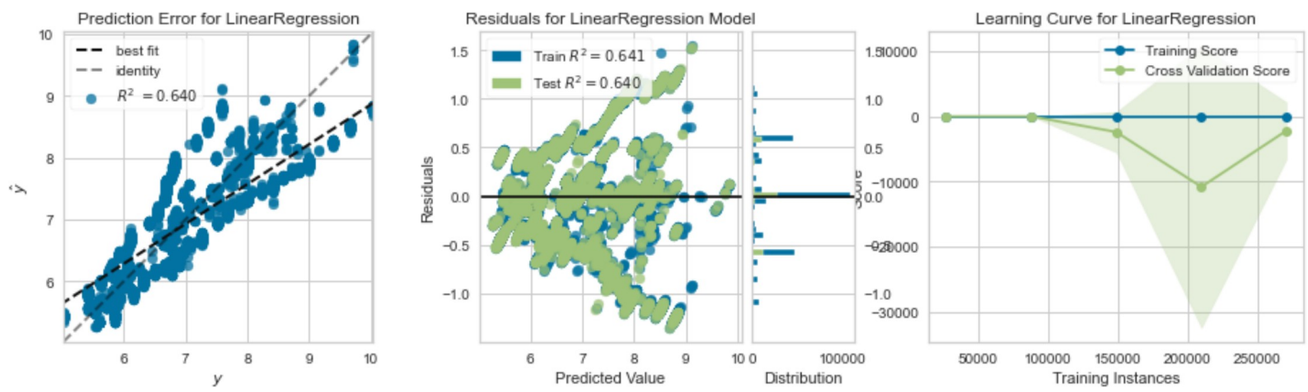
In [102…
```
print('Test set evaluation:\n')
print_evaluate_results(y_test,y_pred_test)
print('======')
print('Train set evaluation:\n')
print_evaluate_results(y_train,y_pred_train)
print('======')
print('CV model evaluation:\n')
print_cross_val(lin_reg)
```

```
Test set evaluation:

MSE: 0.24864 RMSE: 0.49863 R2_Squared 0.63994
======
Train set evaluation:

MSE: 0.24989 RMSE: 0.49989 R2_Squared 0.64138
======
CV model evaluation:

CV Results: Mean: 0.50016 Std: 0.00118 Model R2_Squared 0.6411
```

In [103…
```
#Get the statistics model summary
model1 = pd.DataFrame(data = [['Linear',\
*evaluate_model(y_test,y_pred_test), \
cross_val(lin_reg)[0],cross_val(lin_reg)[1],\
                cross_val(lin_reg)[2]]],
    columns=['Model','MSE','RMSE', 'R Squared', \
    'CV MSE mean', 'CV MSE std', 'Model score'])
print(model1)
```

```
    Model       MSE      RMSE  R Squared  CV MSE mean  CV MSE std  Model score
0  Linear  0.248636  0.498634   0.639939     0.500162    0.001177     0.641096
```

In [104…
```
#ElasticNet Regression Plots
fig, ax = plt.subplots(ncols=3, figsize=(16,4))
visualizer1 = PredictionError(lin_reg,ax=ax[0])
visualizer1.fit(X_train, y_train)
visualizer1.score(X_test, y_test)
visualizer1.finalize()
visualizer2 = ResidualsPlot(lin_reg,ax=ax[1])
visualizer2.fit(X_train, y_train)
visualizer2.score(X_test, y_test)
visualizer2.finalize()
visualizer3 = LearningCurve(lin_reg,ax=ax[2])
visualizer3.fit(X_train, y_train)
visualizer3.finalize()
```



In [105…
```
reg_pred = lin_reg.predict(Pred_data)
result1 = pd.DataFrame()
# exp(x) - 1, the inverse of log1p
result1['Point_Price1'] = np.exp(reg_pred)-1
#Output the result from the linear model
result1
```

Out[105]:

	Point_Price1
0	946.917403
1	2964.696113
2	1194.917147
3	1198.158216
4	863.213921
...	...
335999	594.210212
336000	594.210212
336001	594.690920
336002	594.690920
336003	2529.037095

336004 rows × 1 columns

In [106…
```
result1.describe()
```

Out[106]:

	Point_Price1
count	336004.000000
mean	2782.623211
std	2071.240275
min	255.826233
25%	1060.642906
50%	2217.267432
75%	3870.402730
max	15944.680205

Ridge Regression

In [107…
```
#Fit Ridge regression model and turn the model to
#obtain best alpha value and the best model score
ridge=Ridge()
params= {'alpha': [0.0005,0.001,0.1, 1, 10]}
ridge_grid=GridSearchCV(ridge, param_grid=params,\
cv=kfold,scoring='neg_mean_squared_error', n_jobs=-1)
ridge_grid.fit(X_train,y_train)
alpha = ridge_grid.best_params_
ridge_score = ridge_grid.best_score_
ridge_reg=Ridge(alpha=alpha['alpha'])
ridge_reg.fit(X_train,y_train)
y_predrid_train=ridge_reg.predict(X_train)
y_predrid_test=ridge_reg.predict(X_test)
print("The best alpha value is:",alpha['alpha'],'with score:',ridge_score)
```

```
The best alpha value is: 1 with score: -0.2506006624493399
```

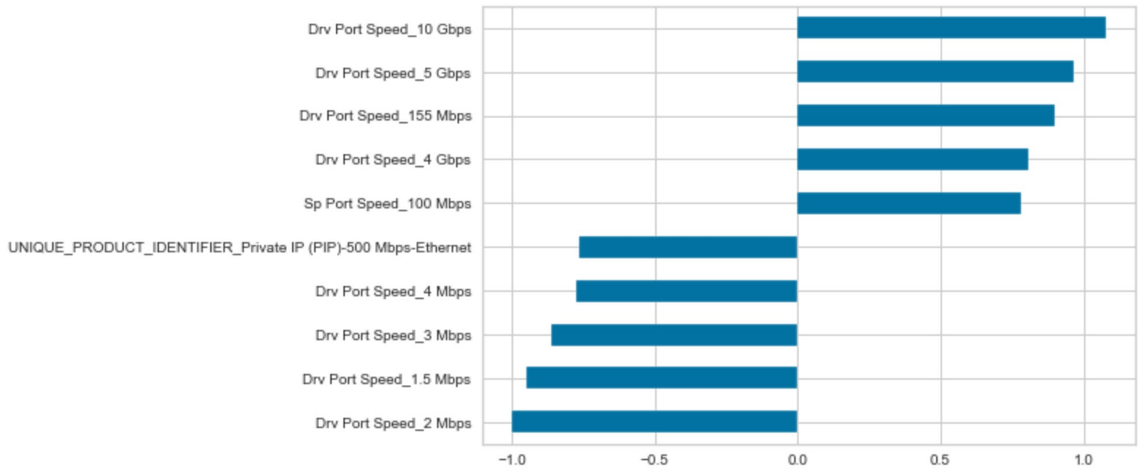
In [108…
```
coeff_df = pd.DataFrame(ridge_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df.sort_values(['Coefficient'], ascending=False)
```

Out[108]:

	Coefficient
Drv Port Speed_10 Gbps	1.076565
Drv Port Speed_5 Gbps	0.967146
Drv Port Speed_155 Mbps	0.898516
Drv Port Speed_4 Gbps	0.808746
Sp Port Speed_100 Mbps	0.780441
...	...
UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-500 Mbps-Ethernet	-0.765690
Drv Port Speed_4 Mbps	-0.774691
Drv Port Speed_3 Mbps	-0.864014
Drv Port Speed_1.5 Mbps	-0.951427
Drv Port Speed_2 Mbps	-1.001517

475 rows × 1 columns

In [109…
```
en_coefs = pd.Series(ridge_reg.coef_, index = X_train.columns)
imp_coefs = pd.concat([en_coefs.sort_values().head(),\
(en_coefs[en_coefs==0]),en_coefs.sort_values().tail()])
imp_coefs.plot(kind = "barh")
plt.show()
```



In [110…
```
ridge_predss = pd.Series(y_predrid_test)
ridge_predss
```

Out[110]:
```
0        6.786357
1        6.669836
2        6.227128
3        5.982513
4        7.969820
           ...
84618    5.790295
84619    7.419362
84620    7.430286
84621    6.683110
84622    6.705789
Length: 84623, dtype: float64
```

In [111…
```
#To get a list of alpha with corresponding mean test scores
res = pd.DataFrame(ridge_grid.cv_results_['params'])
res['score'] = ridge_grid.cv_results_['mean_test_score']
res.sort_values('score',ascending=False).head(3)
```

Out[111]:

	alpha	score
3	1.000	-0.250601
2	0.100	-0.250619
1	0.001	-0.250624

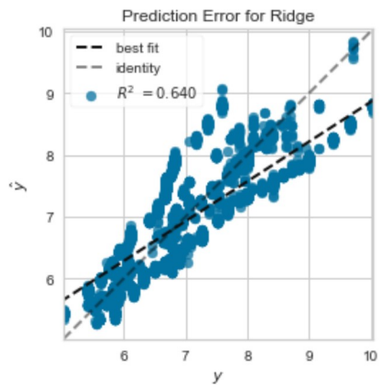
In [112…
```
print('Test set evaluation:\n')
print_evaluate_results(y_test,y_predrid_test)
print('======')
print('Train set evaluation:\n')
print_evaluate_results(y_train,y_predrid_train)
print('======')
print('CV model evaluation:\n')
print_cross_val(ridge_reg)
```

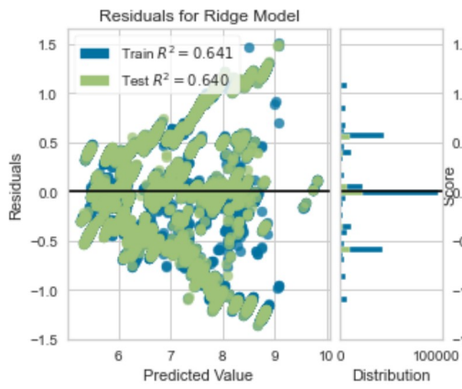
```
Test set evaluation:

MSE: 0.24862 RMSE: 0.49862 R2_Squared 0.63996
======
Train set evaluation:

MSE: 0.24989 RMSE: 0.49989 R2_Squared 0.64137
======
CV model evaluation:

CV Results: Mean: 0.50014 Std: 0.00118 Model R2_Squared 0.64109
```

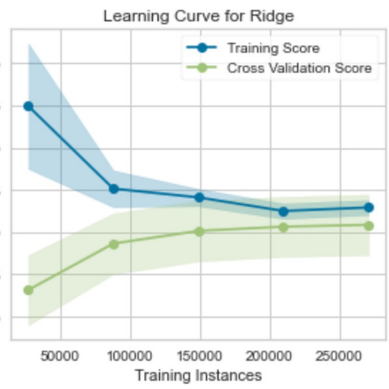
In [113…
```
#Get the statistics model summary
model2 = pd.DataFrame(data = [['Ridge',\
*evaluate_model(y_test,y_predrid_test), \
cross_val(ridge_reg)[0],cross_val(ridge_reg)[1],\
               cross_val(ridge_reg)[2]]],
    columns=['Model','MSE','RMSE', 'R Squared', \
    'CV MSE mean', 'CV MSE std', 'Model score'])
print(model2)
```

```
   Model       MSE      RMSE  R Squared  CV MSE mean  CV MSE std  Model score
0  Ridge  0.248621  0.498619    0.63996      0.50014     0.00118     0.641092
```

In [114…
```
#Ridge Regression Analysis Plots
fig, ax = plt.subplots(ncols=3, figsize=(16,4))
visualizer1 = PredictionError(ridge_reg,ax=ax[0])
visualizer1.fit(X_train, y_train)
visualizer1.score(X_test, y_test)
visualizer1.finalize()
visualizer2 = ResidualsPlot(ridge_reg,ax=ax[1])
visualizer2.fit(X_train, y_train)
visualizer2.score(X_test, y_test)
visualizer2.finalize()
visualizer3 = LearningCurve(ridge_reg,ax=ax[2])
visualizer3.fit(X_train, y_train)
visualizer3.finalize()
```







In [115…
```
rid_pred = ridge_reg.predict(Pred_data)
result2 = pd.DataFrame()

# exp(x) - 1, the inverse of log1p
result2['Point_Price2'] = np.exp(rid_pred)-1
#Output the result from the ridge model
result2
```

Out[115]:

	Point_Price2
0	949.299372
1	2916.566877
2	1177.248264
3	1180.451385
4	878.159421
…	…
335999	601.008352
336000	601.008352
336001	601.492805
336002	601.492805
336003	2609.977575

336004 rows × 1 columns

Lasso Regression

In [116…
```
#Fit Lasso regression model and turn the model to
#obtain best alpha value and the best model score
lasso_reg =Lasso()
params= {'alpha': [0.001,0.1,1]}
lasso_grid = GridSearchCV(lasso_reg, param_grid=params,\
cv=kfold,scoring='neg_mean_squared_error', n_jobs=-1)
lasso_grid.fit(X_train,y_train)
alpha = lasso_grid.best_params_
lasso_score = lasso_grid.best_score_
lasso_reg=Lasso(alpha=alpha['alpha'])
lasso_reg.fit(X_train,y_train)
y_predlas_train=lasso_reg.predict(X_train)
y_predlas_test=lasso_reg.predict(X_test)
print("The best alpha value is:",alpha['alpha'],'with score:',lasso_score)
```

```
The best alpha value is: 0.001 with score: -0.2633881409338161
```

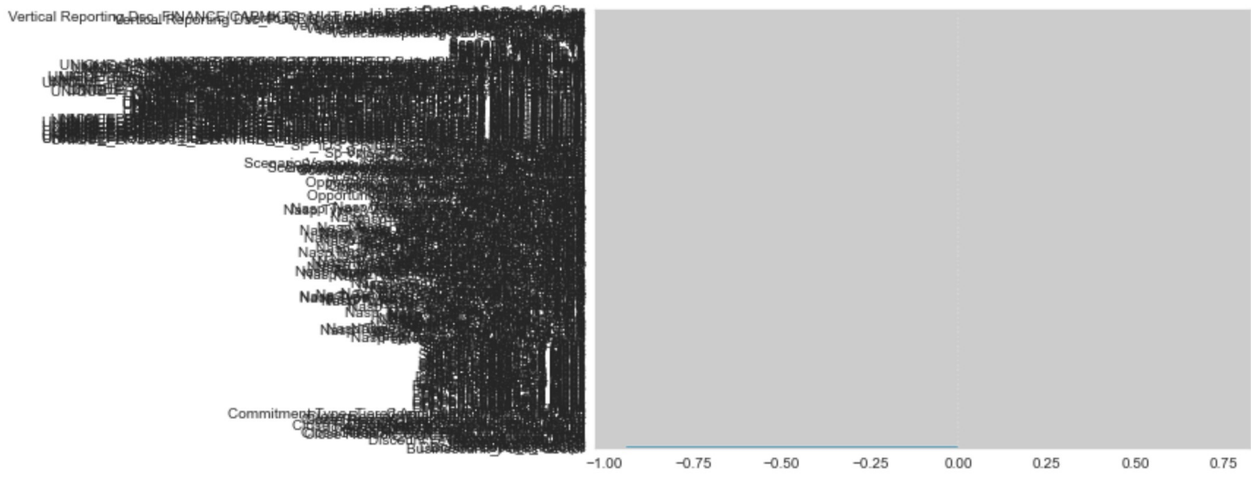
In [117…
```
coeff_df = pd.DataFrame(lasso_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df.sort_values(['Coefficient'], ascending=False)
```

Out[117]:

	Coefficient
Drv Port Speed_10 Gbps	0.737926
UNIQUE_PRODUCT_IDENTIFIER_Internet Dedicated Services-1 Gbps-Ethernet	0.547702
Drv Port Speed_300 Mbps	0.539527
Sp Port Speed_200 Mbps	0.452398
Sp Port Speed_100 Mbps	0.408841
…	…
Drv Port Speed_5 Mbps	-0.595613
Drv Port Speed_3 Mbps	-0.608439
Drv Port Speed_4 Mbps	-0.647336
Drv Port Speed_2 Mbps	-0.932141
Drv Port Speed_1 Mbps	-0.942404

475 rows × 1 columns

In [118…
```
en_coefs = pd.Series(lasso_reg.coef_, index = X_train.columns)
imp_coefs = pd.concat([en_coefs.sort_values().head(1),\
(en_coefs[en_coefs==0]),en_coefs.sort_values().tail(1)])
imp_coefs.plot(kind = "barh")
plt.show()
```



In [119…
```
lasso_predss = pd.Series(y_predlas_test)
lasso_predss
```

Out[119]:
```
0        6.844268
1        6.602778
2        6.176444
3        5.966267
4        8.052048
           ...
84618    5.798129
84619    7.448273
84620    7.403792
84621    6.666293
84622    6.754693
Length: 84623, dtype: float64
```

In [120…
```
#To get a list of alpha with corresponding mean test scores
res = pd.DataFrame(lasso_grid.cv_results_['params'])
res['score'] = lasso_grid.cv_results_['mean_test_score']
res.sort_values('score',ascending=False).head(3)
```

Out[120]:

	alpha	score
0	0.001	-0.263388
1	0.100	-0.525226
2	1.000	-0.696307

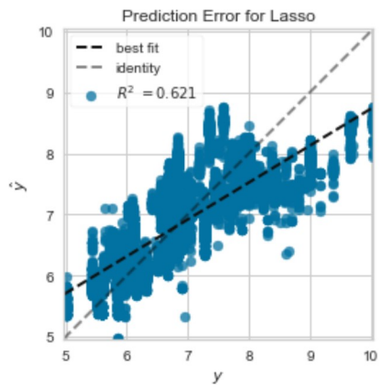
In [121…
```
print('Test set evaluation:\n')
print_evaluate_results(y_test,y_predlas_test)
print('======')
print('Train set evaluation:\n')
print_evaluate_results(y_train,y_predlas_train)
print('======')
print('CV model evaluation:\n')
print_cross_val(lasso_reg)
```

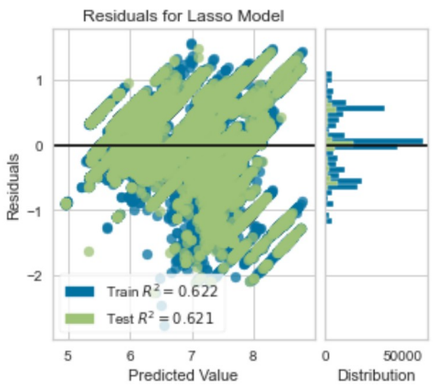
```
Test set evaluation:

MSE: 0.26171 RMSE: 0.51157 R2_Squared 0.62101
======
Train set evaluation:

MSE: 0.26324 RMSE: 0.51307 R2_Squared 0.62222
======
CV model evaluation:

CV Results: Mean: 0.51285 Std: 0.00087 Model R2_Squared 0.62198
```

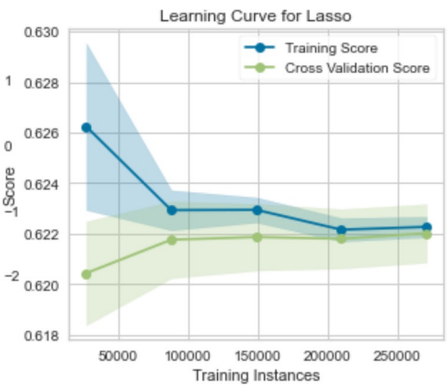
In [122…
```
#Get the statistics model summary
model3 = pd.DataFrame(data = [['Lasso',*evaluate_model\
(y_test,y_predlas_test), cross_val(lasso_reg)[0],\
cross_val(lasso_reg)[1],cross_val(lasso_reg)[2]]],\
columns=['Model','MSE','RMSE', 'R Squared',\
'CV MSE mean', 'CV MSE std', 'Model score'])
print(model3)
```

```
   Model       MSE      RMSE  R Squared  CV MSE mean  CV MSE std  Model score
0  Lasso  0.261707  0.511573    0.62101     0.512848    0.000868     0.621983
```

In [123…
```
#Lasso Regression Analysis Plots
fig, ax = plt.subplots(ncols=3, figsize=(16,4))
visualizer1 = PredictionError(lasso_reg,ax=ax[0])
visualizer1.fit(X_train, y_train)
visualizer1.score(X_test, y_test)
visualizer1.finalize()
visualizer2 = ResidualsPlot(lasso_reg,ax=ax[1])
visualizer2.fit(X_train, y_train)
visualizer2.score(X_test, y_test)
visualizer2.finalize()
visualizer3 = LearningCurve(lasso_reg,ax=ax[2])
visualizer3.fit(X_train, y_train)
visualizer3.finalize()
```







In [124…
```
laso_pred = lasso_reg.predict(Pred_data)
result3 = pd.DataFrame()

# exp(x) - 1, the inverse of log1p
result3['Point_Price3'] = np.exp(laso_pred)-1
#Output the result from the lasso model
result3
```

Out[124]:

	Point_Price3
0	1262.845298
1	2581.676255
2	1236.031664
3	1244.218925
4	1283.336868
…	…
335999	1577.151608
336000	1577.151608
336001	1578.168743
336002	1578.168743
336003	3107.267171

336004 rows × 1 columns

ElasticNet Regression

In [125…
```
#Fit elasticnet regression model and turn the model to
#obtain best estimators and the best model score
elasticnet_reg =ElasticNet()
params= {'alpha': [0.001, 0.01,1],
        'l1_ratio': [0.05,0.5,1]}
elasticnet_grid=GridSearchCV(elasticnet_reg, \
param_grid=params,cv=kfold,scoring='neg_mean_squared_error', n_jobs=-1)
elasticnet_grid.fit(X_train,y_train)
elasticnet_reg = elasticnet_grid.best_estimator_
elasticnet_score = elasticnet_grid.best_score_
elasticnet_reg.fit(X_train,y_train)
y_predent_train=elasticnet_reg.predict(X_train)
y_predent_test=elasticnet_reg.predict(X_test)
print("The best estimator is:",elasticnet_reg, \
      'with score:',elasticnet_score)
```

```
The best estimator is: ElasticNet(alpha=0.001, l1_ratio=0.05) with score: -0.25444827320848973
```

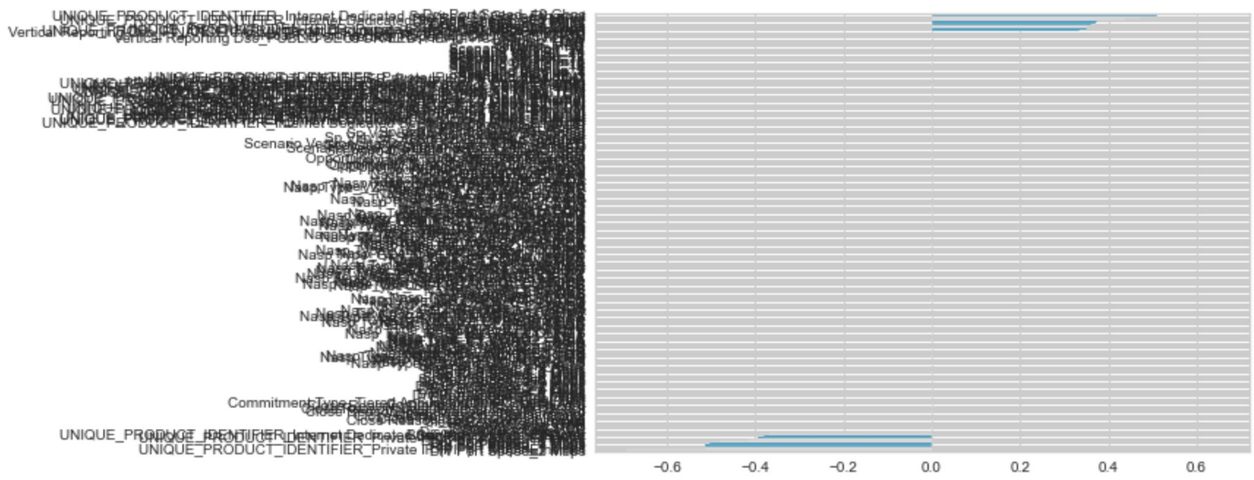
In [126…
```
coeff_df = pd.DataFrame(elasticnet_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df.sort_values(['Coefficient'], ascending=False)
```

Out[126]:

	Coefficient
Drv Port Speed_10 Gbps	0.651768
UNIQUE_PRODUCT_IDENTIFIER_Internet Dedicated Services-10 Gbps-Ethernet	0.509929
Drv Port Speed_300 Mbps	0.474827
UNIQUE_PRODUCT_IDENTIFIER_Internet Dedicated Services-1 Gbps-Ethernet	0.437641
Drv Port Speed_200 Mbps	0.375304
…	…
Drv Port Speed_3 Mbps	-0.503790
Sp Port Speed_1 Mbps	-0.512025
Drv Port Speed_1 Mbps	-0.512260
UNIQUE_PRODUCT_IDENTIFIER_Private IP (PIP)-1 Mbps-Ethernet	-0.514448
Drv Port Speed_2 Mbps	-0.694014

475 rows × 1 columns

In [127…
```
en_coefs = pd.Series(elasticnet_reg.coef_, index = X_train.columns)
imp_coefs = pd.concat([en_coefs.sort_values().head(10),\
(en_coefs[en_coefs==0]),en_coefs.sort_values().tail(10)])
imp_coefs.plot(kind = "barh")
plt.show()
```



In [128…
```
#To get a list of alpha with corresponding mean test scores
res = pd.DataFrame(elasticnet_grid.cv_results_['params'])
res['score'] = elasticnet_grid.cv_results_['mean_test_score']
res.sort_values('score',ascending=False).head(3)
```

Out[128]:

	alpha	l1_ratio	score
0	0.001	0.05	-0.254448
1	0.001	0.50	-0.259006
2	0.001	1.00	-0.263388

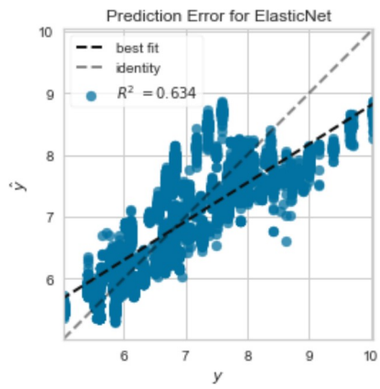
In [129…
```
print('Test set evaluation:\n')
print_evaluate_results(y_test,y_predent_test)
print('======')
print('Train set evaluation:\n')
print_evaluate_results(y_train,y_predent_train)
print('======')
print('CV model evaluation:\n')
print_cross_val(elasticnet_reg)
```

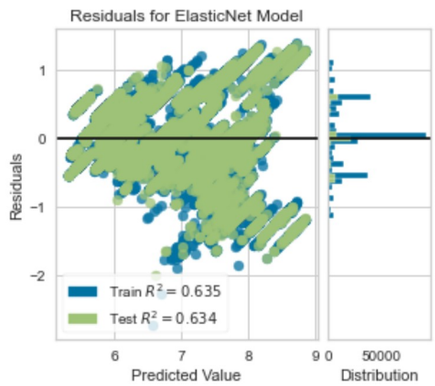
```
Test set evaluation:

MSE: 0.25249 RMSE: 0.50249 R2_Squared 0.63435
======
Train set evaluation:

MSE: 0.25415 RMSE: 0.50413 R2_Squared 0.63526
======
CV model evaluation:

CV Results: Mean: 0.50401 Std: 0.00111 Model R2_Squared 0.63508
```

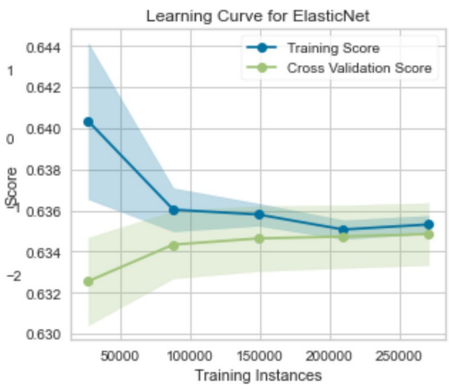
In [130…
```
#Get the statistics model summary
model4 = pd.DataFrame(data = [['ElasticNet',\
*evaluate_model(y_test,y_predent_test), \
cross_val(elasticnet_reg)[0],cross_val\
(elasticnet_reg)[1],cross_val(elasticnet_reg)[2]]],
columns=['Model','MSE','RMSE', 'R Squared', \
'CV MSE mean', 'CV MSE std', 'Model score'])
print(model4)
```

```
        Model       MSE      RMSE  R Squared  CV MSE mean  CV MSE std  \
0  ElasticNet  0.252494  0.502487   0.634352     0.504007     0.00111

   Model score
0     0.635081
```

In [131…
```
#ElasticNet Regression Plots
fig, ax = plt.subplots(ncols=3, figsize=(16,4))
visualizer1 = PredictionError(elasticnet_reg,ax=ax[0])
visualizer1.fit(X_train, y_train)
visualizer1.score(X_test, y_test)
visualizer1.finalize()
visualizer2 = ResidualsPlot(elasticnet_reg,ax=ax[1])
visualizer2.fit(X_train, y_train)
visualizer2.score(X_test, y_test)
visualizer2.finalize()
visualizer3 = LearningCurve(elasticnet_reg,ax=ax[2])
visualizer3.fit(X_train, y_train)
visualizer3.finalize()
```







In [132…
```
Elas_pred = elasticnet_reg.predict(Pred_data)
result4 = pd.DataFrame()

# exp(x) - 1, the inverse of log1p
result4['Point_Price4'] = np.exp(Elas_pred)-1
#Output the result from the elasticnet model
result4
```

Out[132]:

	Point_Price4
0	1211.733869
1	2775.200506
2	1240.391018
3	1246.393447
4	1119.022739
…	…
335999	895.769404
336000	895.769404
336001	896.469770
336002	896.469770
336003	2002.173763

336004 rows × 1 columns

Convolutional Neural Network

In [133…
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

from tensorflow.keras.layers import BatchNormalization
```

In [134…
```
X_train.shape
```

Out[134]:
```
(338488, 475)
```

In [135…
```
from keras.models import Sequential
```

In [136…
```
# Model Build

model = Sequential()
model.add(Conv1D(32, 2, activation="relu", input_shape=(475, 1)))
model.add(Flatten())
model.add(Dense(64, activation="relu"))
model.add(Dense(1))
model.compile(loss="mse", optimizer="adam")

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 474, 32)           96

 flatten (Flatten)           (None, 15168)             0

 dense (Dense)               (None, 64)                970816

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 970,977
Trainable params: 970,977
Non-trainable params: 0
_________________________________________________________________
```

In [137…
```
model.fit(X_train, y_train, batch_size=200,epochs=5, verbose=0)
```

Out[137]:
```
<keras.callbacks.History at 0x1afdaf707f0>
```

In [138…
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score)
```

```
Test loss: 3.8750579357147217
```

In [139…
```
y_predcnn_train=model.predict(X_train)
y_predcnn_test=model.predict(X_test)

print('Test set evaluation:\n')
print_evaluate_results(y_test,y_predcnn_test)
print('======')
print('Train set evaluation:\n')
print_evaluate_results(y_train,y_predcnn_train)
```

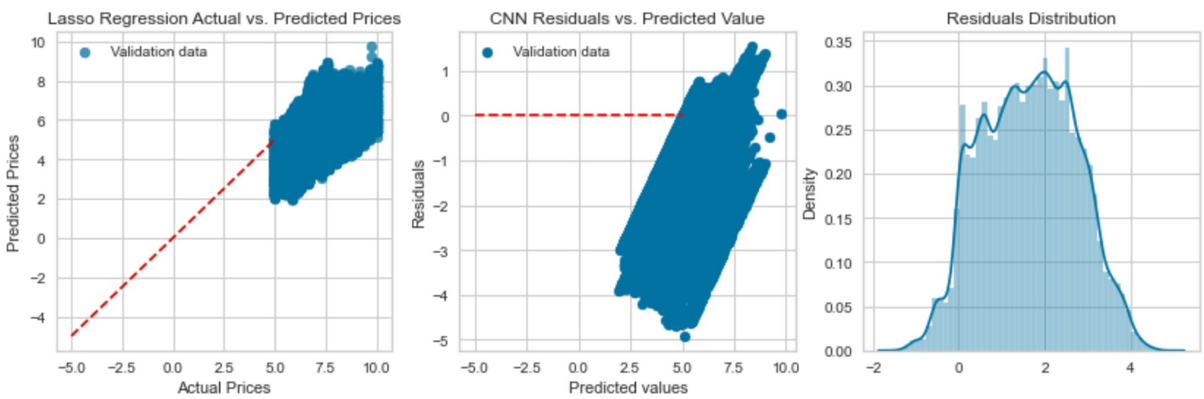
```
Test set evaluation:

MSE: 3.87506 RMSE: 1.96852 R2_Squared -4.61166
======
Train set evaluation:

MSE: 3.85719 RMSE: 1.96397 R2_Squared -4.53553
```

In [140…
```
cnn_pred_test = model.predict_generator(X_test)
```

In [141…
```
cnn_pred_test = pd.Series(cnn_pred_test.flatten())
```

In [142…
```
y_test = y_test.reset_index(drop=True)
```

In [143…
```
#Lasso Regression Model Actual Vs. Predicted Plot
fig, ax = plt.subplots(ncols=3, figsize=(14,4))
ax[0].scatter(y_test, cnn_pred_test,alpha=.75, \
              color='b', label = "Validation data")
ax[0].plot([-5, 5], [-5, 5], '--k', color='r')
ax[0].set_ylabel('Predicted Prices');
ax[0].set_xlabel('Actual Prices');
ax[0].set_title("Lasso Regression Actual vs. Predicted Prices")
ax[0].legend(loc = "upper left")
#Lasso Regression Model Residual Plot Against Predicted Value
resid1 = cnn_pred_test - y_test
ax[1].scatter(y_predcnn_test, resid1, c = "b", \
              label = "Validation data")
ax[1].set_title("CNN Residuals vs. Predicted Value" )
ax[1].set_xlabel("Predicted values")
ax[1].set_ylabel("Residuals")
ax[1].legend(loc = "upper left")
ax[1].hlines(y = 0, xmin = -5, xmax = 5,\
             linestyles='dashed', color = "red")
residuals = y_test - cnn_pred_test
sns.distplot(residuals)
ax[2].set_title("Residuals Distribution" )
plt.show()
```



In [144…
```
#Get the statistics model summary
model5 = pd.DataFrame(data = [['CNN',\
*evaluate_model(y_test,cnn_pred_test),]],
columns=['Model','MSE','RMSE', 'R Squared'])
print(model5)
```

```
  Model       MSE      RMSE  R Squared
0   CNN  3.875059  1.968517  -4.611658
```

In [145…
```
cnn_pred = model.predict(Pred_data)
cnn_pred = pd.Series(cnn_pred.flatten())
result5 = pd.DataFrame()

# exp(x) - 1, the inverse of log1p
result5['Point_Price5'] = np.exp(cnn_pred)-1
#Output the result from the cnn model
result5
```

Out[145]:

	Point_Price5
0	373.737976
1	776.752075
2	29.560045
3	719.821472
4	1325.914185
...	...
335999	203.682922
336000	203.682632
336001	1106.805786
336002	1106.805786
336003	80.537071

336004 rows × 1 columns

In [146…
```
#Define the validation data from original open opportunity point price
Val_data = pd.DataFrame(np.exp(testdata_scaled_digit['Price_log'])-1)
Val_data
```

Out[146]:

	Price_log
0	6000.0
1	2428.0
2	2999.0
3	2999.0
4	3299.0
...	...
335999	1550.0
336000	1550.0
336001	1550.0
336002	1550.0
336003	800.0

336004 rows × 1 columns

In [147…
```
final_result = pd.concat([Val_data,result1, result2,result3, result4,result5], axis=1, join="inner")
```

In [148…
```
final_result.describe().round(1)
```

Out[148]:

	Price_log	Point_Price1	Point_Price2	Point_Price3	Point_Price4	Point_Price5
count	336004.0	336004.0	336004.0	336004.0	336004.0	336004.0
mean	2223.3	2782.6	2585.5	1338.6	1533.8	414.3
std	7430.9	2071.2	1765.9	684.8	698.6	540.8
min	150.0	255.8	269.4	239.9	310.2	4.4
25%	390.0	1060.6	1087.2	841.4	1010.9	72.8
50%	792.0	2217.3	2220.3	1148.2	1411.2	191.6
75%	1431.0	3870.4	3723.8	1792.0	1924.2	547.4
max	72446.0	15944.7	10100.7	5529.4	5119.7	6912.6

In [149…
```
final_result.to_csv('final_result.csv') #save the prediction results to excel
```

Model Evalutation and Comparsion

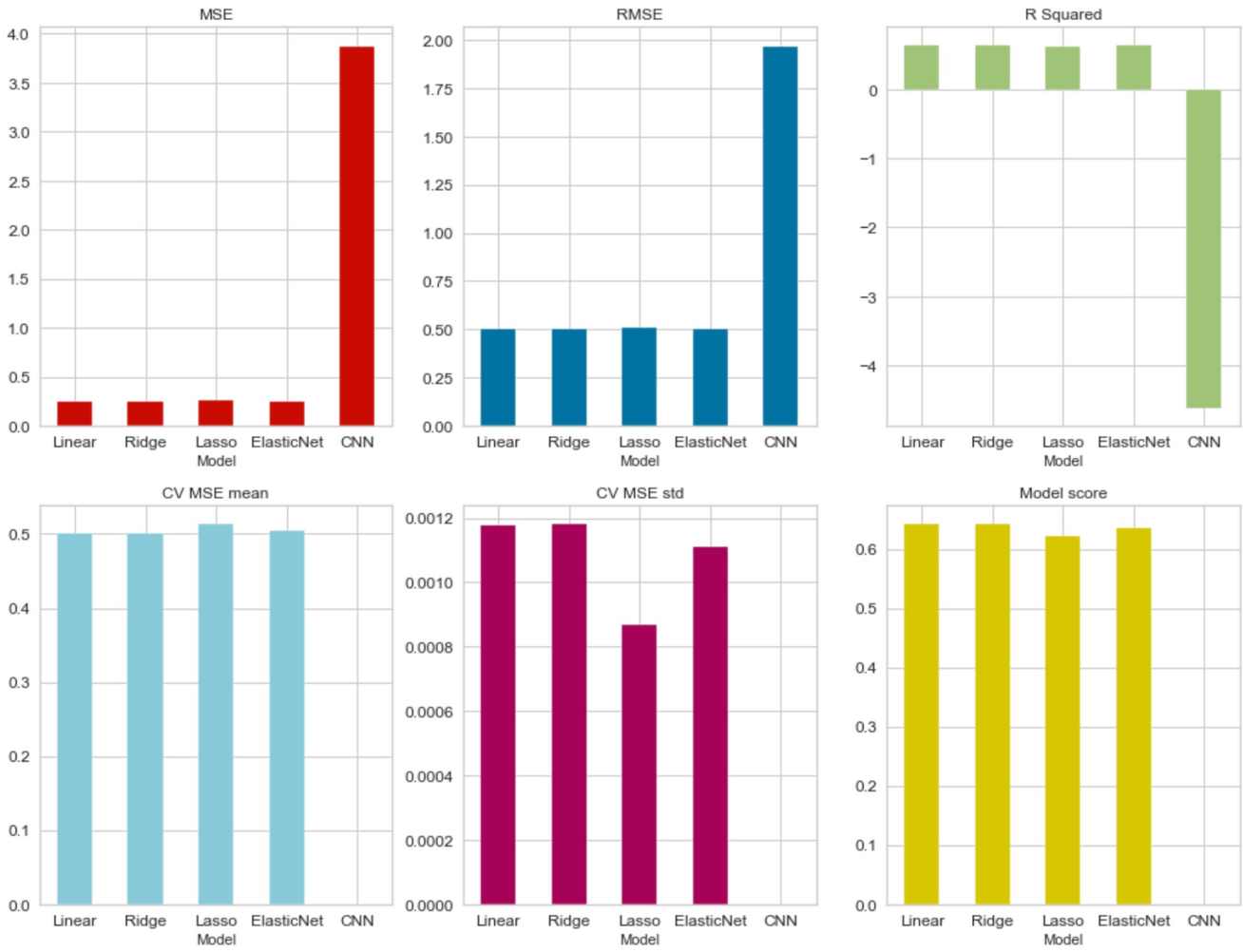
In [150…
```
frames = [model1,model2, model3, model4, model5]
result = pd.concat(frames)
result.set_index("Model", drop = True, inplace = True)
result.sort_values(by=['RMSE'])
```

Out[150]:

Model	MSE	RMSE	R Squared	CV MSE mean	CV MSE std	Model score
Ridge	0.248621	0.498619	0.639960	0.500140	0.001180	0.641092
Linear	0.248636	0.498634	0.639939	0.500162	0.001177	0.641096
ElasticNet	0.252494	0.502487	0.634352	0.504007	0.001110	0.635081
Lasso	0.261707	0.511573	0.621010	0.512848	0.000868	0.621983
CNN	3.875059	1.968517	-4.611658	NaN	NaN	NaN

In [151…

```
#Get the overview distribution of the performance scores
fig, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = \
plt.subplots(2,3,figsize=(16,12))
fig1 = result['MSE'].plot(kind='bar',\
title ="MSE", color = 'r',fontsize=12,ax=ax1)
fig2 = result['RMSE'].plot(kind='bar', \
title ="RMSE", color = 'b', fontsize=12,ax=ax2)
fig3 = result['R Squared'].plot(kind='bar', \
title ="R Squared", color = 'g', fontsize=12,ax=ax3)
fig4 = result['CV MSE mean'].plot(kind='bar', \
title ="CV MSE mean", color = 'c', fontsize=12,ax=ax4)
fig5 = result['CV MSE std'].plot(kind='bar', \
title ="CV MSE std",color = 'm', fontsize=12,ax=ax5)
fig6 = result['Model score'].plot(kind='bar', \
title ="Model score",color = 'y', fontsize=12,ax=ax6)
for ax in fig.axes:
    ax.tick_params(labelrotation=0)
```



Ridge regression tended to give us a slightly better leaderboard score overall because it generates the lowest RMSE(MSE) score and highest R-Squared score which means have more predictive power to explain the variability within this model. Therefore, we decided to use Ridge regression as the final model for this project. In order to get better prediction results and there is room for further tuning and scaling to achieve more accurate prediction results. In the next step, we will also implement more regression and Neural Network models if time permits.