In [1]:

```
!pip install xgboost
!pip install flask
!pip install joblib
```

```
Requirement already satisfied: scipy in c:\users\vincentxd24\anaconda3
\lib\site-packages (from xgboost) (1.7.1)
Requirement already satisfied: numpy in c:\users\vincentxd24\anaconda3
\lib\site-packages (from xgboost) (1.20.3)
Requirement already satisfied: flask in c:\users\vincentxd24\anaconda3
\lib\site-packages (1.1.2)
Requirement already satisfied: click>=5.1 in c:\users\vincentxd24\anaco
nda3\lib\site-packages (from flask) (8.0.3)
Requirement already satisfied: Werkzeug>=0.15 in c:\users\vincentxd24\a
naconda3\lib\site-packages (from flask) (2.0.2)
Requirement already satisfied: itsdangerous>=0.24 in c:\users\vincentxd
24\anaconda3\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\vincentxd24\a
naconda3\lib\site-packages (from flask) (2.11.3)
Requirement already satisfied: colorama in c:\users\vincentxd24\anacond
a3\lib\site-packages (from click>=5.1->flask) (0.4.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\vincentxd24
\anaconda3\lib\site-packages (from Jinja2>=2.10.1->flask) (1.1.1)
Requirement already satisfied: joblib in c:\users\vincentxd24\anaconda3
\lib\site-packages (1.1.0)
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from keras.models import Sequential
from keras.layers import Dense
import xgboost as xgb
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error, r2_score
import pickle
from flask import Flask, jsonify, request
import joblib
```

In [3]:

```
# read csv data
df = pd.read_csv('bus-breakdown-and-delays.csv', low_memory=False)
df.head()
```

Out[3]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
0	2015-2016	1227538	Special Ed AM Run	2621	J711	Heavy Traffic	
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
4	2015-2016	1227542	Special Ed AM Run	3124	M373	Heavy Traffic	

5 rows × 21 columns

In [4]:

```
df.describe()
```

Out[4]:

	Busbreakdown_ID	Number_Of_Students_On_The_Bus
count	2.585900e+04	25859.000000
mean	1.259452e+06	4.122162
std	5.157505e+04	78.305998
min	1.212691e+06	0.000000
25%	1.235498e+06	0.000000
50%	1.247422e+06	0.000000
75%	1.258546e+06	4.000000
max	1.471604e+06	9007.000000

In [5]:

```
df.shape
```

Out[5]:

```
(25859, 21)
```

In [6]:

```
df['Number_Of_Students_On_The_Bus'].value_counts()
```

Out[6]:

```
0       13556
2        1943
3        1781
1        1675
4        1306
        ...
47          1
1492        1
1315        1
1749        1
39          1
Name: Number_Of_Students_On_The_Bus, Length: 62, dtype: int64
```

In [7]:

```
df.dtypes
```

Out[7]:

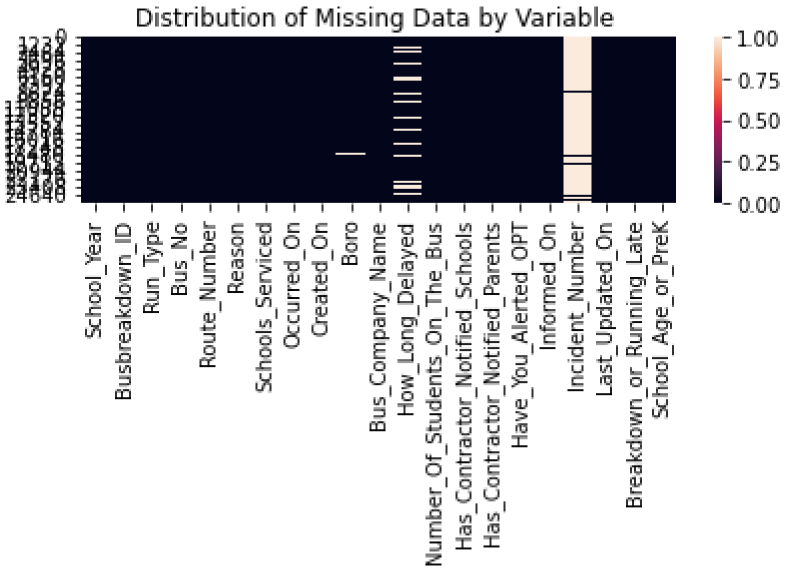
```
School_Year                        object
Busbreakdown_ID                     int64
Run_Type                           object
Bus_No                             object
Route_Number                       object
Reason                             object
Schools_Serviced                   object
Occurred_On                        object
Created_On                         object
Boro                               object
Bus_Company_Name                   object
How_Long_Delayed                   object
Number_Of_Students_On_The_Bus       int64
Has_Contractor_Notified_Schools    object
Has_Contractor_Notified_Parents    object
Have_You_Alerted_OPT               object
Informed_On                        object
Incident_Number                    object
Last_Updated_On                    object
Breakdown_or_Running_Late          object
School_Age_or_PreK                 object
dtype: object
```

In [8]:

```
sns.heatmap(df.isnull()) #See distribution of missing data
plt.figsize = (5,2.5)
plt.tight_layout()
plt.title('Distribution of Missing Data by Variable ')
```

Out[8]:

```
Text(0.5, 1.0, 'Distribution of Missing Data by Variable ')
```



In [9]:

```
# Display details of dataset
print ("Rows :" ,df.shape[0])
print ("Columns :" ,df.shape[1])
print ("\nFeatures :\n" ,df.columns.tolist())
print ("\nMissing values : \n", df.isnull().sum())
print ("\nUnique values : \n",df.nunique())
```

```
Rows : 25859
Columns : 21

Features :
 ['School_Year', 'Busbreakdown_ID', 'Run_Type', 'Bus_No', 'Route_Number',
'Reason', 'Schools_Serviced', 'Occurred_On', 'Created_On', 'Boro', 'Bus_Co
mpany_Name', 'How_Long_Delayed', 'Number_Of_Students_On_The_Bus', 'Has_Con
tractor_Notified_Schools', 'Has_Contractor_Notified_Parents', 'Have_You_Al
erted_OPT', 'Informed_On', 'Incident_Number', 'Last_Updated_On', 'Breakdow
n_or_Running_Late', 'School_Age_or_PreK']

Missing values :
 School_Year                            0
Busbreakdown_ID                        0
Run_Type                               0
Bus_No                                 0
Route_Number                           1
Reason                                 0
Schools_Serviced                       1
Occurred_On                            0
Created_On                             0
Boro                                1042
Bus_Company_Name                       0
How_Long_Delayed                    3782
Number_Of_Students_On_The_Bus          0
Has_Contractor_Notified_Schools        0
Has_Contractor_Notified_Parents        0
Have_You_Alerted_OPT                   0
Informed_On                            0
Incident_Number                    24724
Last_Updated_On                        0
Breakdown_or_Running_Late              0
School_Age_or_PreK                     0
dtype: int64

Unique values :
 School_Year                            4
Busbreakdown_ID                    25848
Run_Type                               9
Bus_No                              6352
Route_Number                        6713
Reason                                10
Schools_Serviced                    3512
Occurred_On                        14643
Created_On                         15152
Boro                                  11
Bus_Company_Name                      99
How_Long_Delayed                     779
Number_Of_Students_On_The_Bus         62
Has_Contractor_Notified_Schools        2
Has_Contractor_Notified_Parents        2
Have_You_Alerted_OPT                   2
Informed_On                        15152
Incident_Number                     1019
Last_Updated_On                    23742
Breakdown_or_Running_Late              2
School_Age_or_PreK                     2
dtype: int64
```

Data Cleaning

In [10]:

```
#Drop incident number, most of column is missing
df = df.drop(['Incident_Number'], axis = 1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25859 entries, 0 to 25858
Data columns (total 20 columns):
 #   Column                            Non-Null Count  Dtype 
---  ------                            --------------  ----- 
 0   School_Year                       25859 non-null  object
 1   Busbreakdown_ID                   25859 non-null  int64 
 2   Run_Type                          25859 non-null  object
 3   Bus_No                            25859 non-null  object
 4   Route_Number                      25858 non-null  object
 5   Reason                            25859 non-null  object
 6   Schools_Serviced                  25858 non-null  object
 7   Occurred_On                       25859 non-null  object
 8   Created_On                        25859 non-null  object
 9   Boro                              24817 non-null  object
 10  Bus_Company_Name                  25859 non-null  object
 11  How_Long_Delayed                  22077 non-null  object
 12  Number_Of_Students_On_The_Bus     25859 non-null  int64 
 13  Has_Contractor_Notified_Schools   25859 non-null  object
 14  Has_Contractor_Notified_Parents   25859 non-null  object
 15  Have_You_Alerted_OPT              25859 non-null  object
 16  Informed_On                       25859 non-null  object
 17  Last_Updated_On                   25859 non-null  object
 18  Breakdown_or_Running_Late         25859 non-null  object
 19  School_Age_or_PreK                25859 non-null  object
dtypes: int64(2), object(18)
memory usage: 3.9+ MB
```

In [11]:

```
print(df.columns)
```

```
Index(['School_Year', 'Busbreakdown_ID', 'Run_Type', 'Bus_No', 'Route_Numb
er',
       'Reason', 'Schools_Serviced', 'Occurred_On', 'Created_On', 'Boro',
       'Bus_Company_Name', 'How_Long_Delayed', 'Number_Of_Students_On_The_
Bus',
       'Has_Contractor_Notified_Schools', 'Has_Contractor_Notified_Parent
s',
       'Have_You_Alerted_OPT', 'Informed_On', 'Last_Updated_On',
       'Breakdown_or_Running_Late', 'School_Age_or_PreK'],
      dtype='object')
```

In [12]:

```
#Extract digits from string column
df.loc[:, 'Delay'] = df['How_Long_Delayed'].str.extract('(\d+)').copy()
#Check if regex worked- Yes!
df.head()
```

Out[12]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
0	2015-2016	1227538	Special Ed AM Run	2621	J711	Heavy Traffic	
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
4	2015-2016	1227542	Special Ed AM Run	3124	M373	Heavy Traffic	

5 rows × 21 columns

In [13]:

```
df[df['Delay'].isnull()]#Check if data is null
#We see that there's question marks or other irregularaties- lets drop this data
```

Out[13]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	
0	2015-2016	1227538	Special Ed AM Run	2621	J711	Heavy Traffic	
4	2015-2016	1227542	Special Ed AM Run	3124	M373	Heavy Traffic	
16	2015-2016	1227558	Special Ed AM Run	2052	L524	Flat Tire	
18	2015-2016	1227561	Special Ed AM Run	2508	L531	Heavy Traffic	
22	2015-2016	1227077	General Ed AM Run	2675	X2189	Other	
...	...	...	...	...	...	...	
25828	2015-2016	1264153	Special Ed AM Run	TN0408	L599	Mechanical Problem	
25835	2015-2016	1215164	Pre-K/EI	1009	WOC#7	Mechanical Problem	
25837	2015-2016	1216543	Special Ed AM Run	2423	M198	Heavy Traffic	
25853	2017-2018	1429429	Special Ed AM Run	540D	Q777	Mechanical Problem	
25854	2017-2018	1429431	Special Ed AM Run	5343	K315	Mechanical Problem	2210

3898 rows × 21 columns

In [14]:

```
df_clean = df.dropna() #Drop remainaing NAs
df_clean.info()
df_clean.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21072 entries, 1 to 25858
Data columns (total 21 columns):
 #   Column                           Non-Null Count  Dtype 
---  ------                           --------------  ----- 
 0   School_Year                      21072 non-null  object
 1   Busbreakdown_ID                  21072 non-null  int64 
 2   Run_Type                         21072 non-null  object
 3   Bus_No                           21072 non-null  object
 4   Route_Number                     21072 non-null  object
 5   Reason                           21072 non-null  object
 6   Schools_Serviced                 21072 non-null  object
 7   Occurred_On                      21072 non-null  object
 8   Created_On                       21072 non-null  object
 9   Boro                             21072 non-null  object
 10  Bus_Company_Name                 21072 non-null  object
 11  How_Long_Delayed                 21072 non-null  object
 12  Number_Of_Students_On_The_Bus    21072 non-null  int64 
 13  Has_Contractor_Notified_Schools  21072 non-null  object
 14  Has_Contractor_Notified_Parents  21072 non-null  object
 15  Have_You_Alerted_OPT             21072 non-null  object
 16  Informed_On                      21072 non-null  object
 17  Last_Updated_On                  21072 non-null  object
 18  Breakdown_or_Running_Late        21072 non-null  object
 19  School_Age_or_PreK               21072 non-null  object
 20  Delay                            21072 non-null  object
dtypes: int64(2), object(19)
memory usage: 3.5+ MB
```

Out[14]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
5	2015-2016	1227543	Special Ed AM Run	HT1502	W796	Heavy Traffic	
6	2015-2016	1227544	Special Ed AM Run	142	W633	Heavy Traffic	

5 rows × 21 columns

In [15]:

```
df_clean = df_clean.dropna() #Drop new NAs
df_clean.isnull().sum() #Check that no NAs are left
```

Out[15]:

```
School_Year                          0
Busbreakdown_ID                      0
Run_Type                             0
Bus_No                               0
Route_Number                         0
Reason                               0
Schools_Serviced                     0
Occurred_On                          0
Created_On                           0
Boro                                 0
Bus_Company_Name                     0
How_Long_Delayed                     0
Number_Of_Students_On_The_Bus        0
Has_Contractor_Notified_Schools      0
Has_Contractor_Notified_Parents      0
Have_You_Alerted_OPT                 0
Informed_On                          0
Last_Updated_On                      0
Breakdown_or_Running_Late            0
School_Age_or_PreK                   0
Delay                                0
dtype: int64
```

In [16]:

```
#Convert string to integer
df_clean.loc[:, 'Delay'] = pd.to_numeric(df_clean['Delay'])
```

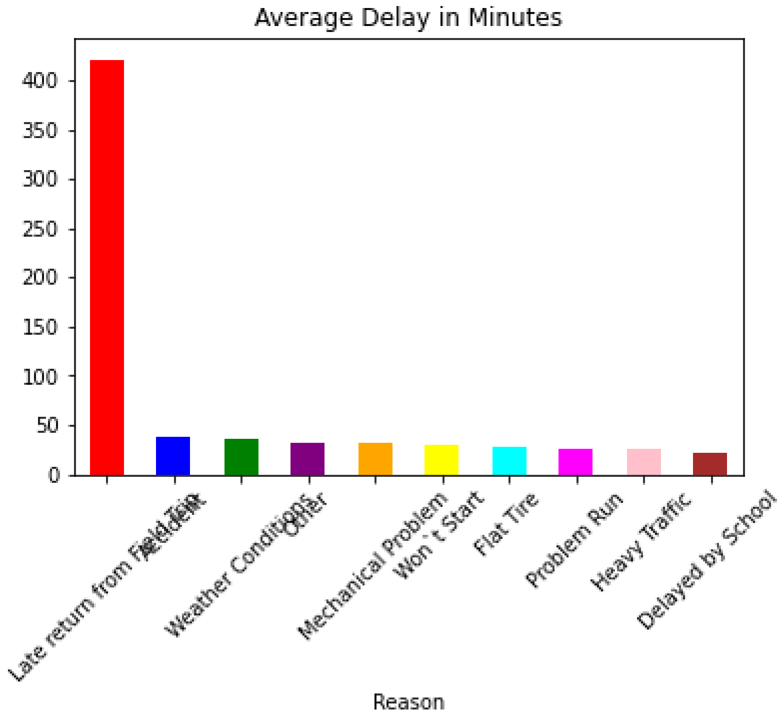
In [17]:

```
#Drop original column
df_clean = df_clean.drop(['How_Long_Delayed'], axis = 1)
```

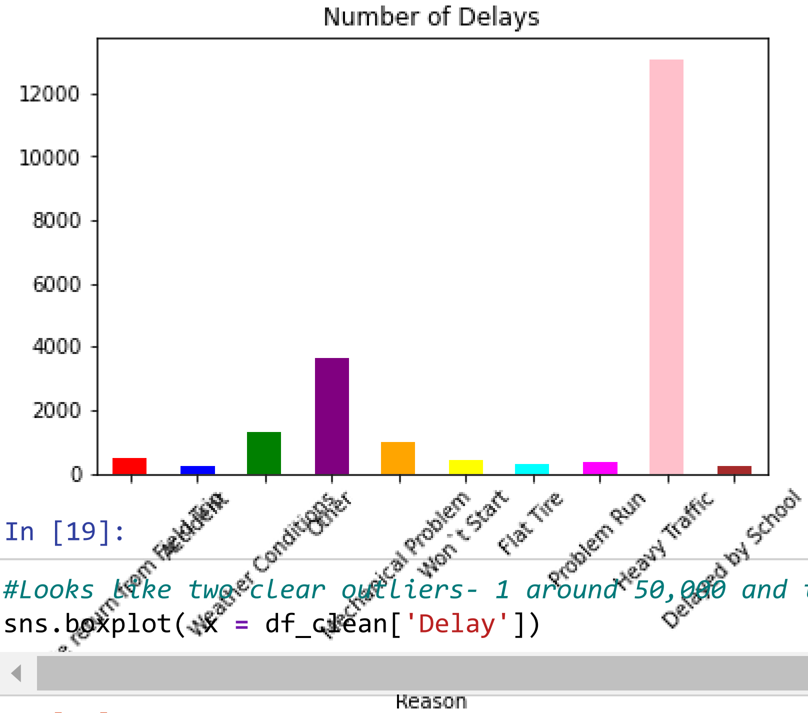
In [18]:

```
reasons = pd.pivot_table(df_clean, index = 'Reason', values = 'Delay', aggfunc = [np.mea


# Define a list of 10 different colors
colors = ['red', 'blue', 'green', 'purple', 'orange', 'yellow', 'cyan', 'magenta', 'pink

# Create a pivot table of reasons and delay
reasons = pd.pivot_table(df_clean, index='Reason', values='Delay', aggfunc=[np.mean, np.

# Plot the average delay chart
plt.figure(figsize=(12, 6))
reasons.plot(kind='bar', y=('mean', 'Delay'), color=colors)
plt.title('Average Delay in Minutes')
plt.xticks(rotation=45)
plt.legend().remove()
plt.show()

# Plot the number of delays chart
plt.figure(figsize=(10, 6))
reasons.plot(kind='bar', y=('size', 'Delay'), color=colors)
plt.title('Number of Delays')
plt.xticks(rotation=45)
plt.legend().remove()
plt.show()
```

```
<Figure size 864x432 with 0 Axes>
```




```
<Figure size 720x432 with 0 Axes>
```

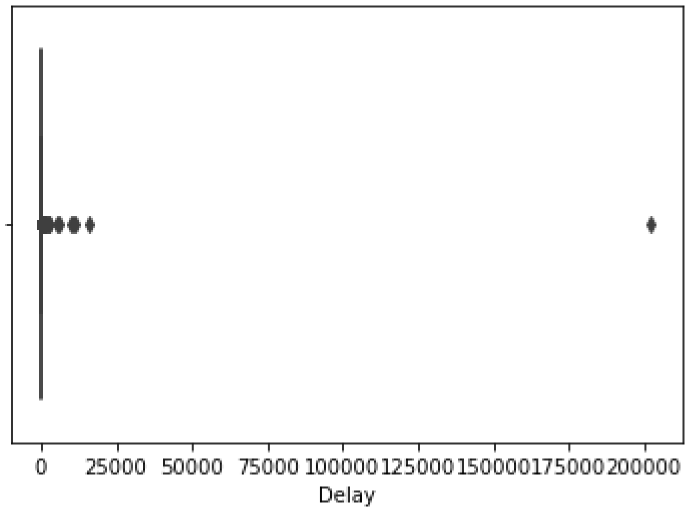


In [19]:

```
#Looks like two clear outliers- 1 around 50,000 and the other around 200,000 Let's remov
sns.boxplot(x = df_clean['Delay'])
```

Out[19]:

```
<AxesSubplot:xlabel='Delay'>
```

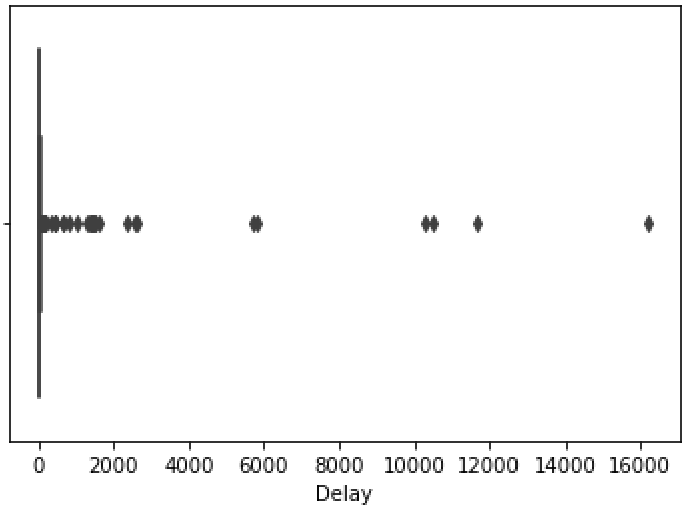


In [20]:

```
df_exoutliers = df_clean[df_clean['Delay'] < 50000]
sns.boxplot(x = df_exoutliers['Delay']) #Check if we need to remove further outliers
```

Out[20]:

```
<AxesSubplot:xlabel='Delay'>
```



In [21]:

```
df_clean.head()
```

Out[21]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
5	2015-2016	1227543	Special Ed AM Run	HT1502	W796	Heavy Traffic	
6	2015-2016	1227544	Special Ed AM Run	142	W633	Heavy Traffic	

In [22]:

```
df_clean['Route_Number'].value_counts()
```

Out[22]:

```
1        301
3        249
2        236
4        176
5        170
        ...
Q994       1
R1050      1
R1141      1
R1233      1
K8468      1
Name: Route_Number, Length: 5724, dtype: int64
```

In [23]:

```
pd.pivot_table(df_clean, index = 'Route_Number',
               values = 'Delay',
               aggfunc = [np.mean,np.size]).sort_values(by = ('size','Delay'),
                                                        ascending = False).head(6)
```

Out[23]:

	mean	size
	Delay	Delay
Route_Number		
1	17.099668	301
3	19.586345	249
2	17.199153	236
4	16.681818	176
5	19.358824	170
6	19.730000	100

In [24]:

```
#Filter to see cases where route is top 6 in # of delays
routes = ['1','2','3','5','4','6']
top_routes = df_clean[df_clean['Route_Number'].isin(routes)]
```

In [25]:

```
routes_pivot = pd.pivot_table(top_routes,
                              index = 'Route_Number',
                              values = 'Delay',
                              aggfunc = [np.mean,np.size])
routes_pivot.head(6)
```

Out[25]:

	mean	size
	Delay	Delay
Route_Number		
1	17.099668	301
2	17.199153	236
3	19.586345	249
4	16.681818	176
5	19.358824	170
6	19.730000	100

In [26]:

```
df_clean.head()
```

Out[26]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
5	2015-2016	1227543	Special Ed AM Run	HT1502	W796	Heavy Traffic	
6	2015-2016	1227544	Special Ed AM Run	142	W633	Heavy Traffic	

In [27]:

```
df_clean['Bus_Company_Name'].value_counts()
```

Out[27]:

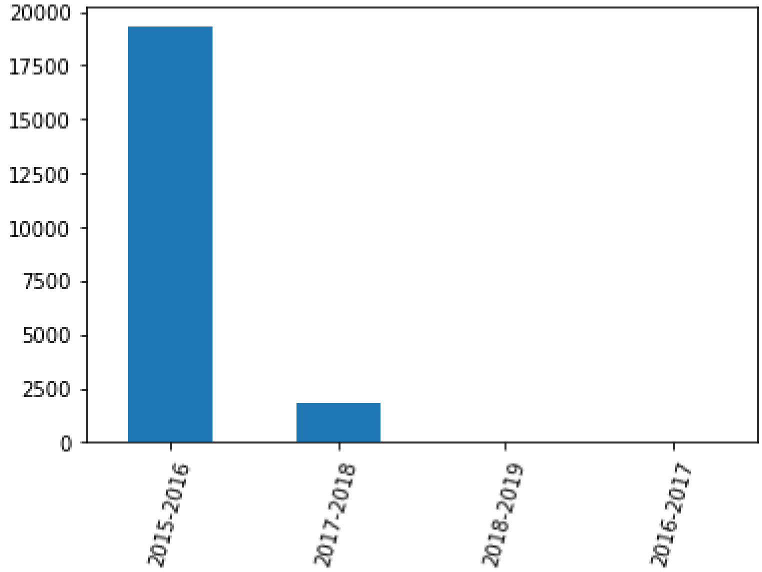
```
PIONEER TRANSPORTATION CO     1992
LEESEL TRANSP CORP (B2192     1833
NEW DAWN TRANSIT, LLC (B2     1694
G.V.C., LTD.                  1660
RELIANT TRANS, INC. (B232     1652
                              ...
THIRD AVENUE TRANSIT, INC        1
ALL COUNTY BUS LLC (B2321)       1
Y & M TRANSIT CORP (B2321        1
MONTAUK STUDENT TRANS LLC        1
Y & M TRANSIT CORP (B2321)       1
Name: Bus_Company_Name, Length: 99, dtype: int64
```

In [28]:

```
#First Let's remove unnecessary features, checking 1 by 1

#School Year- is it relevant?
df_clean['School_Year'].value_counts().plot(kind = 'bar')
plt.xticks(rotation = 75) #Make Data cleaner to read

#See an increasing trend year on year in quantity-let's investigate if there's any signi
```

Out[28]:

```
(array([0, 1, 2, 3]),
 [Text(0, 0, '2015-2016'),
  Text(1, 0, '2017-2018'),
  Text(2, 0, '2018-2019'),
  Text(3, 0, '2016-2017')])
```



In [29]:

```
#Let's first see average delay, across the dataset
df_clean['Delay'].mean() #Around 29 mins is the average delay time
```

Out[29]:

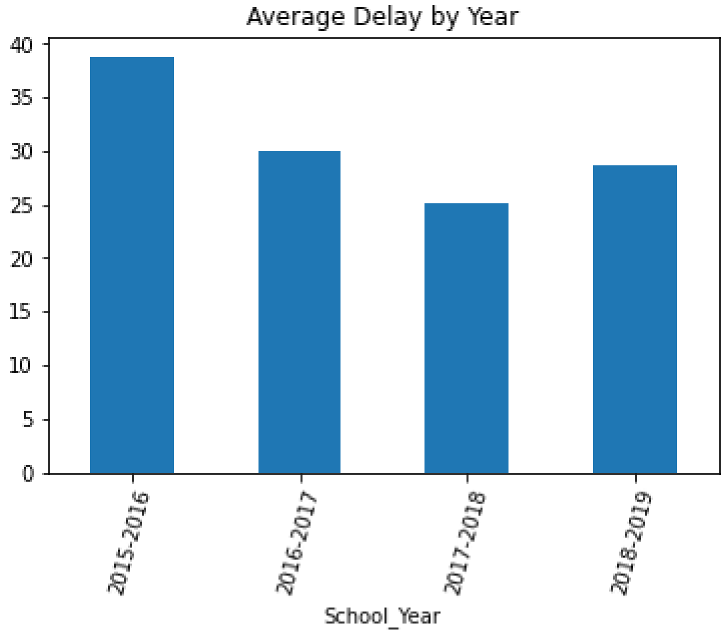
```
37.542663249810175
```

In [30]:

```
pd.pivot_table(df_clean, index = 'School_Year', values = 'Delay', aggfunc = np.mean).plo
plt.legend().remove() #Get rid of legend
plt.title('Average Delay by Year')
plt.xticks(rotation = 75) #Make easier to read

#Doesn't look any year is terribly far off from another but also not congruent- will kee
```

Out[30]:

```
(array([0, 1, 2, 3]),
 [Text(0, 0, '2015-2016'),
  Text(1, 0, '2016-2017'),
  Text(2, 0, '2017-2018'),
  Text(3, 0, '2018-2019')])
```



In [31]:

```
df_clean.head() #Let's check what the data looked like again
```

Out[31]:

	School_Year	Busbreakdown_ID	Run_Type	Bus_No	Route_Number	Reason	Schools_Ser
1	2015-2016	1227539	Special Ed AM Run	1260	M351	Heavy Traffic	
2	2015-2016	1227540	Pre-K/EI	418	3	Heavy Traffic	
3	2015-2016	1227541	Special Ed AM Run	4522	M271	Heavy Traffic	
5	2015-2016	1227543	Special Ed AM Run	HT1502	W796	Heavy Traffic	
6	2015-2016	1227544	Special Ed AM Run	142	W633	Heavy Traffic	

In [32]:

```
#Data seems like no noise, we'll drop
df_clean['Busbreakdown_ID'].value_counts()
```

Out[32]:

```
1243405    2
1243668    2
1243407    2
1227386    2
1427107    2
          ..
1241522    1
1241521    1
1241515    1
1241513    1
1264190    1
Name: Busbreakdown_ID, Length: 21064, dtype: int64
```

In [33]:

```
df_clean = df_clean.drop(['Busbreakdown_ID'], axis = 1)
df_clean.head()
```

Out[33]:

	School_Year	Run_Type	Bus_No	Route_Number	Reason	Schools_Serviced	Occurred_On
1	2015-2016	Special Ed AM Run	1260	M351	Heavy Traffic	6716	2015-11-05 8:10:00
2	2015-2016	Pre-K/EI	418	3	Heavy Traffic	C445	2015-11-05 8:09:00
3	2015-2016	Special Ed AM Run	4522	M271	Heavy Traffic	2699	2015-11-05 8:12:00
5	2015-2016	Special Ed AM Run	HT1502	W796	Heavy Traffic	75407	2015-11-05 7:58:00
6	2015-2016	Special Ed AM Run	142	W633	Heavy Traffic	75670	2015-11-05 8:24:00

In [34]:

```
bus_num = pd.pivot_table(df_clean, index = 'Bus_No', values = 'Delay',aggfunc = np.size)

bus_num

#Create pivot to see number of delays by bus number
#We see that a lot have only have 1.
#Instead of one hot encoding, let's just convert to digits
```

Out[34]:

Bus_No	Delay
9	68
1389	65
213	62
357	60
1836	59
...	...
6620	1
6623	1
25586	1
664	1
44123	1

5579 rows × 1 columns

In [35]:

```
#Extract digits from string column
df_clean['Bus_Number'] = df_clean['Bus_No'].str.extract('(\d+)')
#Convert string to integer
df_clean['Bus_Number'] =  pd.to_numeric(df_clean['Bus_Number'])
df_clean.isnull().sum()
#We now have some more NAs- let's do a quick investigation
```

Out[35]:

```
School_Year                        0
Run_Type                           0
Bus_No                             0
Route_Number                       0
Reason                             0
Schools_Serviced                   0
Occurred_On                        0
Created_On                         0
Boro                               0
Bus_Company_Name                   0
Number_Of_Students_On_The_Bus      0
Has_Contractor_Notified_Schools    0
Has_Contractor_Notified_Parents    0
Have_You_Alerted_OPT               0
Informed_On                        0
Last_Updated_On                    0
Breakdown_or_Running_Late          0
School_Age_or_PreK                 0
Delay                              0
Bus_Number                        12
dtype: int64
```

In [36]:

```
#Looks like noisy data, will drop
df_clean[df_clean['Bus_Number'].isnull()]
df_clean = df_clean.dropna()
#Drop original column
df_clean = df_clean.drop(['Bus_No'], axis = 1)
```

In [37]:

```
df_clean.head()
```

Out[37]:

	School_Year	Run_Type	Route_Number	Reason	Schools_Serviced	Occurred_On	Createc
1	2015-2016	Special Ed AM Run	M351	Heavy Traffic	6716	2015-11-05 8:10:00	2015-' 8:'
2	2015-2016	Pre-K/EI	3	Heavy Traffic	C445	2015-11-05 8:09:00	2015-' 8:'
3	2015-2016	Special Ed AM Run	M271	Heavy Traffic	2699	2015-11-05 8:12:00	2015-' 8:'
5	2015-2016	Special Ed AM Run	W796	Heavy Traffic	75407	2015-11-05 7:58:00	2015-' 8:'
6	2015-2016	Special Ed AM Run	W633	Heavy Traffic	75670	2015-11-05 8:24:00	2015-' 8:'

In [38]:

```
#Let's look at the current correlation across features
df_clean.corr()
```

Out[38]:

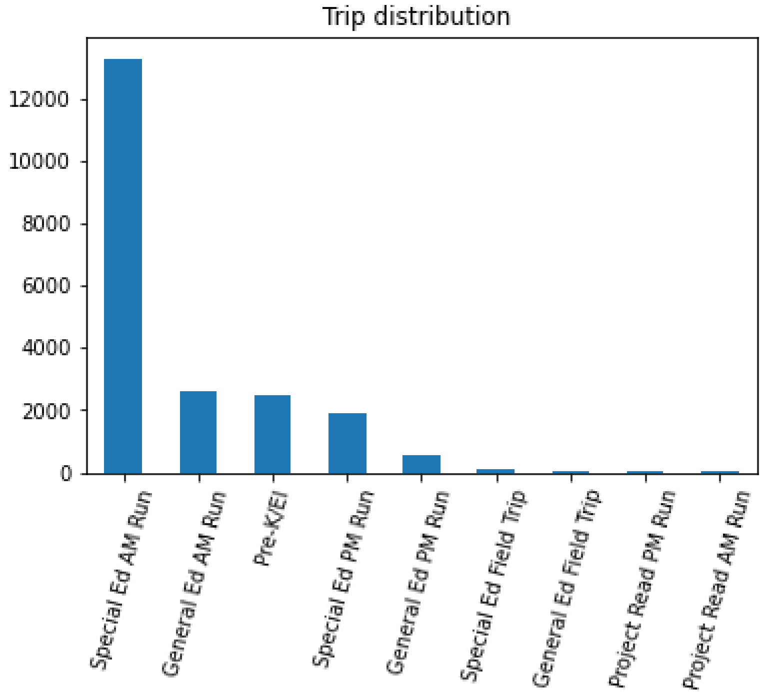
	Number_Of_Students_On_The_Bus	Delay	Bus_Numbei
Number_Of_Students_On_The_Bus	1.000000	-0.000548	-0.001031
Delay	-0.000548	1.000000	-0.00029ç
Bus_Number	-0.001031	-0.000299	1.00000C

In [39]:

```
df_clean['Run_Type'].value_counts().plot(kind = 'bar')
plt.title('Trip distribution ')
plt.xticks(rotation = 75) #Data heavily weighted towards Special Ed AM in terms of quant
```

Out[39]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Special Ed AM Run'),
  Text(1, 0, 'General Ed AM Run'),
  Text(2, 0, 'Pre-K/EI'),
  Text(3, 0, 'Special Ed PM Run'),
  Text(4, 0, 'General Ed PM Run'),
  Text(5, 0, 'Special Ed Field Trip'),
  Text(6, 0, 'General Ed Field Trip'),
  Text(7, 0, 'Project Read PM Run'),
  Text(8, 0, 'Project Read AM Run')])
```



In [40]:

```
df_clean.head(10)
```

Out[40]:

	School_Year	Run_Type	Route_Number	Reason	Schools_Serviced	Occurred_On	Create
1	2015-2016	Special Ed AM Run	M351	Heavy Traffic	6716	2015-11-05 8:10:00	2015 8
2	2015-2016	Pre-K/EI	3	Heavy Traffic	C445	2015-11-05 8:09:00	2015 8
3	2015-2016	Special Ed AM Run	M271	Heavy Traffic	2699	2015-11-05 8:12:00	2015 8
5	2015-2016	Special Ed AM Run	W796	Heavy Traffic	75407	2015-11-05 7:58:00	2015 8
6	2015-2016	Special Ed AM Run	W633	Heavy Traffic	75670	2015-11-05 8:24:00	2015 8
7	2015-2016	Special Ed AM Run	M678	Heavy Traffic	3417	2015-11-05 8:15:00	2015 8
8	2015-2016	Special Ed AM Run	M126	Heavy Traffic	1450	2015-11-05 7:55:00	2015 8
9	2015-2016	Special Ed AM Run	M922	Heavy Traffic	2930	2015-11-05 8:16:00	2015 8
10	2015-2016	Special Ed AM Run	M490	Heavy Traffic	3004	2015-11-05 8:19:00	2015 8
11	2015-2016	Pre-K/EI	10	Heavy Traffic	C601	2015-11-05 8:19:00	2015 8

In [41]:

```
df_clean.nunique()
```

Out[41]:

```
School_Year                          4
Run_Type                             9
Route_Number                      5724
Reason                              10
Schools_Serviced                  3060
Occurred_On                      12478
Created_On                       12924
Boro                                11
Bus_Company_Name                    99
Number_Of_Students_On_The_Bus       60
Has_Contractor_Notified_Schools      2
Has_Contractor_Notified_Parents      2
Have_You_Alerted_OPT                 2
Informed_On                      12924
Last_Updated_On                  19328
Breakdown_or_Running_Late            2
School_Age_or_PreK                   2
Delay                               79
Bus_Number                        4288
dtype: int64
```

Selected Features 1

In [42]:

```
from sklearn.model_selection import train_test_split

y = df_clean['Delay'] #store target variable
X = df_clean[['School_Year','Run_Type','Reason','Boro','Bus_Company_Name','Number_Of_Stu
             'School_Age_or_PreK']] #Added bus company name/school year features
dummy_df = pd.get_dummies(X) #Convert data to dummies to enable modeling
print(dummy_df.shape)
print(y.shape)
```

```
(21060, 138)
(21060,)
```

In [43]:

```
from sklearn.model_selection import train_test_split

X_train, X_test,y_train, y_test = train_test_split(dummy_df,y,test_size = .2,
                                                   random_state = 40)
```

Algorithm

Gradient boosted tree

In [44]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 4],
    'learning_rate': [0.01, 0.1],
    'loss': ['ls', 'lad']
}

# Create a GradientBoostingRegressor object
gbr = GradientBoostingRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=gbr, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
gbr_mae1 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
gbr_rmsle1 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
gbr_cv_mae1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
gbr_cv_rmsle1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("Gradient Boosted Tree MAE:", gbr_mae1)
print("Gradient Boosted Tree RMSLE:", gbr_rmsle1)
print("Cross-validated MAE:", -gbr_cv_mae1)
print("Cross-validated RMSLE:", np.sqrt(-gbr_cv_rmsle1))
```

```
{'learning_rate': 0.1, 'loss': 'lad', 'max_depth': 4, 'n_estimators': 100}
Gradient Boosted Tree MAE: 11.449102788882989
Gradient Boosted Tree RMSLE: 0.6678114345568975
Cross-validated MAE: 24.525587782958326
Cross-validated RMSLE: 0.6765254639058241
```

Multi-layer Perceptron (MLP)

In [45]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the parameter grid to search over
param_grid = {
    'hidden_layer_sizes': [(10,), (50,), (100,)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'sgd'],
    'learning_rate': ['constant', 'adaptive']
}

# Create a MLPRegressor object
mlp = MLPRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
mlp_mae1 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
mlp_rmsle1 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
mlp_cv_mae1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
mlp_cv_rmsle1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("Multi-layer Perceptron MAE:", mlp_mae1)
print("Multi-layer Perceptron RMSLE:", mlp_rmsle1)
print("Cross-validated MAE:", -mlp_cv_mae1)
print("Cross-validated RMSLE:", np.sqrt(-mlp_cv_rmsle1))
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(

{'activation': 'tanh', 'hidden_layer_sizes': (10,), 'learning_rate': 'adap
tive', 'solver': 'adam'}
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(
```

```
Multi-layer Perceptron MAE: 12.506735737946727
Multi-layer Perceptron RMSLE: 0.6870849604076169
Cross-validated MAE: 25.510107232399836
Cross-validated RMSLE: nan
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network\_m
ultilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Neural Network

In [46]:

```
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the neural network model
def create_nn_model():
    model = Sequential()
    model.add(Dense(100, input_shape=(X_train.shape[1],), activation='relu'))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create a KerasRegressor object
nn = KerasRegressor(build_fn=create_nn_model, epochs=50, batch_size=32, verbose=0)

# Use cross-validation to evaluate the model
nn_cv_mae1 = np.mean(cross_val_score(nn, X_train, y_train, cv=5, scoring='neg_mean_absol
nn_cv_rmsle1 = np.mean(cross_val_score(nn, X_train, y_train, cv=5, scoring='neg_mean_squ

# Fit the model to the training data
nn.fit(X_train, y_train)

# Use the model to make predictions on the testing set
y_pred = nn.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
nn_mae1 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
nn_rmsle1 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

print("Neural Network MAE:", nn_mae1)
print("Neural Network RMSLE:", nn_rmsle1)
print("Cross-validated MAE:", -nn_cv_mae1)
print("Cross-validated RMSLE:", np.sqrt(-nn_cv_rmsle1))
```

```
C:\Users\VINCEN~1\AppData\Local\Temp/ipykernel_22012/4249823482.py:15: Dep
recationWarning: KerasRegressor is deprecated, use Sci-Keras (https://gith
ub.com/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/s
table/migration.html (https://www.adriangb.com/scikeras/stable/migration.h
tml) for help migrating.
  nn = KerasRegressor(build_fn=create_nn_model, epochs=50, batch_size=32,
verbose=0)
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
```

```
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(
```

```
Neural Network MAE: 19.774649564130808
Neural Network RMSLE: 0.7920814160128057
Cross-validated MAE: 37.74936490800899
Cross-validated RMSLE: nan
```

XGBoost

In [47]:

```
from xgboost import XGBRegressor

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 4],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.5, 0.8]
}

# Create an XGBRegressor object
xgb = XGBRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
xgb_mae1 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
xgb_rmsle1 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
xgb_cv_mae1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
xgb_cv_rmsle1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("XGBoost MAE:", xgb_mae1)
print("XGBoost RMSLE:", xgb_rmsle1)
print("Cross-validated MAE:", -xgb_cv_mae1)
print("Cross-validated RMSLE:", np.sqrt(-xgb_cv_rmsle1))
```

```
{'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100, 'subsample':
0.8}
XGBoost MAE: 16.111946259015873
XGBoost RMSLE: 0.7273175343242436
Cross-validated MAE: 34.92098090122447
Cross-validated RMSLE: 0.742319734243465
```

ADA Boosting

In [48]:

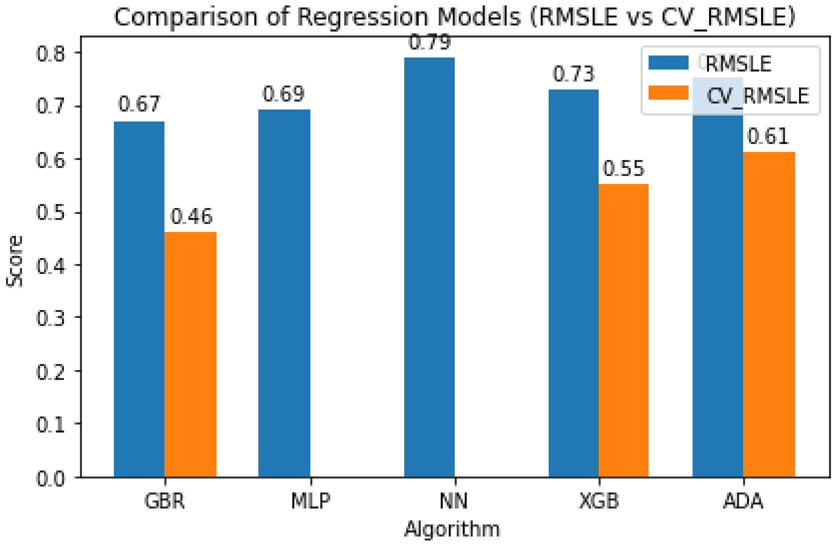
```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1],
    'loss': ['linear', 'square', 'exponential']
}

# Create an AdaBoostRegressor object
ada = AdaBoostRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=ada, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
ada_mae1 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
ada_rmsle1 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
ada_cv_mae1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
ada_cv_rmsle1 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("ADA Boosting MAE:", ada_mae1)
print("ADA Boosting RMSLE:", ada_rmsle1)
print("Cross-validated MAE:", -ada_cv_mae1)
print("Cross-validated RMSLE:", np.sqrt(-ada_cv_rmsle1))
```

```
{'learning_rate': 0.01, 'loss': 'exponential', 'n_estimators': 50}
ADA Boosting MAE: 21.153609863587988
ADA Boosting RMSLE: 0.7527256596366793
Cross-validated MAE: 54.50089624166792
Cross-validated RMSLE: 0.7793397301689164
```

Evaluate the reseult

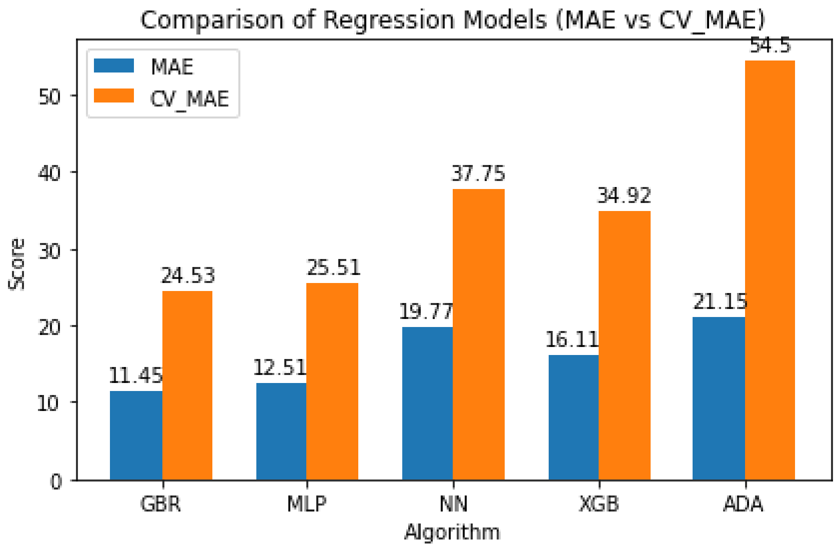
In [60]:

```
# Data for the bar graph
labels = ['GBR', 'MLP', 'NN', 'XGB', 'ADA']
rmsle_scores = [round(gbr_rmsle1, 2), round(mlp_rmsle1, 2), round(nn_rmsle1, 2), round(x
cv_rmsle_scores = [round(-gbr_cv_rmsle1, 2), round(-mlp_cv_rmsle1, 2), round(-nn_cv_rmsl

# Set up the bar graph
x = np.arange(len(labels))
width = 0.35
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, rmsle_scores, width, label='RMSLE')
rects2 = ax.bar(x + width/2, cv_rmsle_scores, width, label='CV_RMSLE')

# Add labels and title
ax.set_xlabel('Algorithm')
ax.set_ylabel('Score')
ax.set_title('Comparison of Regression Models (RMSLE vs CV_RMSLE)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Function to add labels to the bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels to the bars
autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```


Comparison of Regression Models (RMSLE vs CV_RMSLE)
RMSLE
CV_RMSLE
0.67
0.46
0.69
0.79
0.73
0.55
0.61
Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
GBR
MLP
NN
XGB
ADA
Algorithm

In [59]:

```
# Data for the bar graph
labels = ['GBR', 'MLP', 'NN', 'XGB', 'ADA']
mae_scores = [round(gbr_mae1, 2), round(mlp_mae1, 2), round(nn_mae1, 2), round(xgb_mae1,
cv_mae_scores = [round(-gbr_cv_mae1, 2), round(-mlp_cv_mae1, 2), round(-nn_cv_mae1, 2),

# Set up the bar graph
x = np.arange(len(labels))
width = 0.35
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, mae_scores, width, label='MAE')
rects2 = ax.bar(x + width/2, cv_mae_scores, width, label='CV_MAE')

# Add labels and title
ax.set_xlabel('Algorithm')
ax.set_ylabel('Score')
ax.set_title('Comparison of Regression Models (MAE vs CV_MAE)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Function to add labels to the bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels to the bars
autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```



Selected Features 2

In [52]:

```
y = df_clean['Delay'] #store target variable
X = df_clean[['School_Year','Run_Type','Reason','Boro','Bus_Company_Name','Number_Of_Stu
             'School_Age_or_PreK','Schools_Serviced', 'Has_Contractor_Notified_Parents',
    #added additional features related to contractors
dummy_df = pd.get_dummies(X) #look familiar?
X_train, X_test,y_train, y_test = train_test_split(dummy_df,y,test_size = .2, random_sta
```

Algorithm

Gradient boosted tree

In [53]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 4],
    'learning_rate': [0.01, 0.1],
    'loss': ['ls', 'lad']
}

# Create a GradientBoostingRegressor object
gbr = GradientBoostingRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=gbr, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
gbr_mae2 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
gbr_rmsle2 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
gbr_cv_mae2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
gbr_cv_rmsle2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("Gradient Boosted Tree MAE:", gbr_mae2)
print("Gradient Boosted Tree RMSLE:", gbr_rmsle2)
print("Cross-validated MAE:", -gbr_cv_mae2)
print("Cross-validated RMSLE:", np.sqrt(-gbr_cv_rmsle2))
```

```
{'learning_rate': 0.1, 'loss': 'lad', 'max_depth': 4, 'n_estimators': 200}
Gradient Boosted Tree MAE: 11.326174798209209
Gradient Boosted Tree RMSLE: 0.6589585273541234
Cross-validated MAE: 24.51503279505364
Cross-validated RMSLE: 0.6772960388685977
```

Multi-layer Perceptron (MLP)

In [54]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the parameter grid to search over
param_grid = {
    'hidden_layer_sizes': [(10,), (50,), (100,)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'sgd'],
    'learning_rate': ['constant', 'adaptive']
}

# Create a MLPRegressor object
mlp = MLPRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
mlp_mae2 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
mlp_rmsle2 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
mlp_cv_mae2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
mlp_cv_rmsle2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("Multi-layer Perceptron MAE:", mlp_mae2)
print("Multi-layer Perceptron RMSLE:", mlp_rmsle2)
print("Cross-validated MAE:", -mlp_cv_mae2)
print("Cross-validated RMSLE:", np.sqrt(-mlp_cv_rmsle2))
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network
\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converg
ed yet.
  warnings.warn(

{'activation': 'tanh', 'hidden_layer_sizes': (10,), 'learning_rate': 'c
onstant', 'solver': 'adam'}
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network
\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converg
ed yet.
  warnings.warn(
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\neural_network
\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converg
ed yet.
```

Neural Network

In [55]:

```
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the neural network model
def create_nn_model():
    model = Sequential()
    model.add(Dense(100, input_shape=(X_train.shape[1],), activation='relu'))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create a KerasRegressor object
nn = KerasRegressor(build_fn=create_nn_model, epochs=50, batch_size=32, verbose=0)

# Use cross-validation to evaluate the model
nn_cv_mae2 = np.mean(cross_val_score(nn, X_train, y_train, cv=5, scoring='neg_mean_absol
nn_cv_rmsle2 = np.mean(cross_val_score(nn, X_train, y_train, cv=5, scoring='neg_mean_squ

# Fit the model to the training data
nn.fit(X_train, y_train)

# Use the model to make predictions on the testing set
y_pred = nn.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
nn_mae2 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
nn_rmsle2 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

print("Neural Network MAE:", nn_mae2)
print("Neural Network RMSLE:", nn_rmsle2)
print("Cross-validated MAE:", -nn_cv_mae2)
print("Cross-validated RMSLE:", np.sqrt(-nn_cv_rmsle2))
```

```
2, verbose=0)
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selectio
n\_validation.py:696: UserWarning: Scoring failed. The score on this tr
ain-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_
selection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metric
s\_scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metric
s\_scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils
\validation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metric
s\_regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used whe
```

XGBoost

In [56]:

```
from xgboost import XGBRegressor

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 4],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.5, 0.8]
}

# Create an XGBRegressor object
xgb = XGBRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
xgb_mae2 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
xgb_rmsle2 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
xgb_cv_mae2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
xgb_cv_rmsle2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("XGBoost MAE:", xgb_mae2)
print("XGBoost RMSLE:", xgb_rmsle2)
print("Cross-validated MAE:", -xgb_cv_mae2)
print("Cross-validated RMSLE:", np.sqrt(-xgb_cv_rmsle2))
```

```
{'learning_rate': 0.01, 'max_depth': 4, 'n_estimators': 100, 'subsample':
0.8}
```

```
C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:696: UserWarning: Scoring failed. The score on this train-te
st partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 687, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 87, in __call__
    score = scorer._score(cached_call, estimator,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\utils\val
idation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\vincentxd24\anaconda3\lib\site-packages\sklearn\metrics\_
regression.py", line 413, in mean_squared_log_error
    raise ValueError("Mean Squared Logarithmic Error cannot be used when "
ValueError: Mean Squared Logarithmic Error cannot be used when targets con
tain negative values.

  warnings.warn(

XGBoost MAE: 14.198917940924083
XGBoost RMSLE: 0.7242592735292439
Cross-validated MAE: 27.60929808629711
Cross-validated RMSLE: nan
```

ADA Boosting

In [57]:

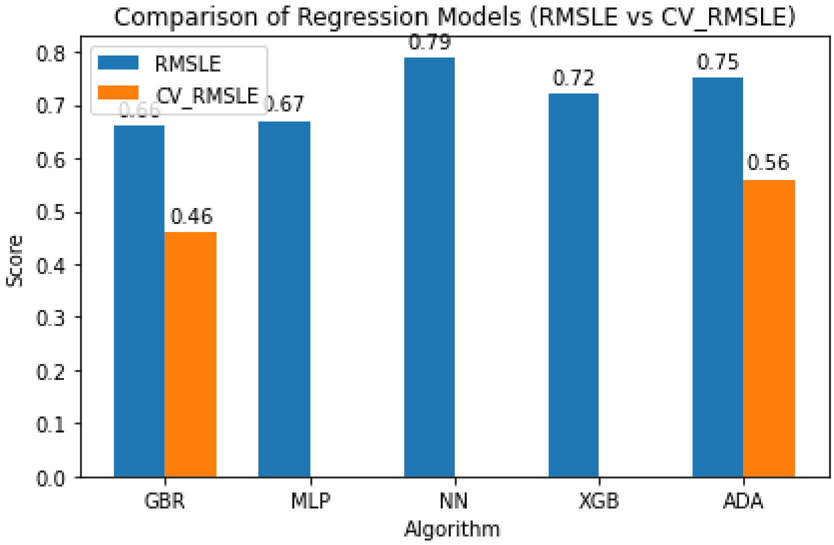
```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_log_error

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [50, 100],
    'learning_rate': [0.01, 0.1],
    'loss': ['linear', 'exponential']
}

# Create an AdaBoostRegressor object
ada = AdaBoostRegressor()

# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=ada, param_grid=param_grid, cv=5, n_jobs=-1)

# Fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)

# Use the best estimator to make predictions on the testing set
y_pred = grid_search.best_estimator_.predict(X_test)

# Calculate the mean absolute error (MAE) and root mean squared logarithmic error (RMSLE
ada_mae2 = mean_absolute_error(y_test, y_pred)

# Calculate the RMSLE, with absolute value transformation for negative values
y_test_abs = np.abs(y_test)
y_pred_abs = np.abs(y_pred)
ada_rmsle2 = np.sqrt(mean_squared_log_error(y_test_abs, y_pred_abs))

# Calculate the cross-validated MAE and RMSLE scores
ada_cv_mae2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=
ada_cv_rmsle2 = np.mean(cross_val_score(grid_search.best_estimator_, X_train, y_train, c

print("ADA Boosting MAE:", ada_mae2)
print("ADA Boosting RMSLE:", ada_rmsle2)
print("Cross-validated MAE:", -ada_cv_mae2)
print("Cross-validated RMSLE:", np.sqrt(-ada_cv_rmsle2))
```

```
{'learning_rate': 0.01, 'loss': 'exponential', 'n_estimators': 50}
ADA Boosting MAE: 14.250693251275221
ADA Boosting RMSLE: 0.7506815991187478
Cross-validated MAE: 27.128558149643652
Cross-validated RMSLE: 0.7504512056272551
```

Evaluate the reseult

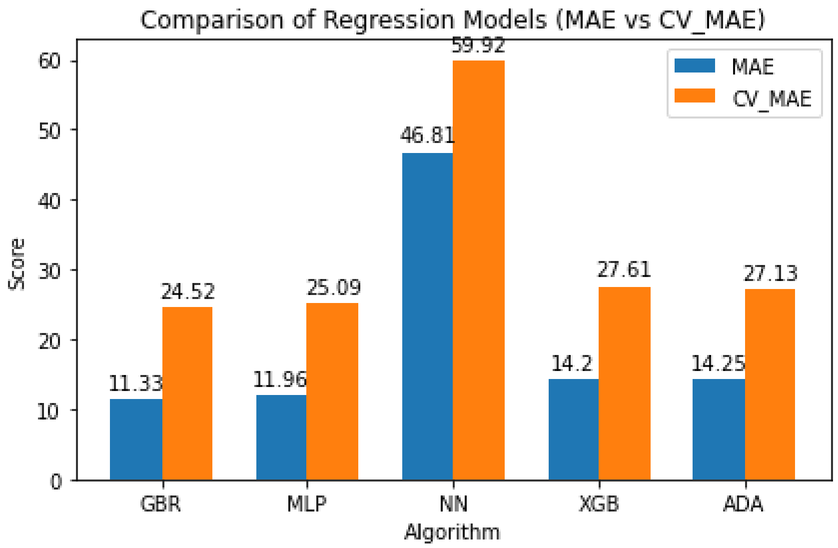
In [61]:

```
# Data for the bar graph
labels = ['GBR', 'MLP', 'NN', 'XGB', 'ADA']
rmsle_scores = [round(gbr_rmsle2, 2), round(mlp_rmsle2, 2), round(nn_rmsle1, 2), round(x
cv_rmsle_scores = [round(-gbr_cv_rmsle2, 2), round(-mlp_cv_rmsle2, 2), round(-nn_cv_rmsl

# Set up the bar graph
x = np.arange(len(labels))
width = 0.35
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, rmsle_scores, width, label='RMSLE')
rects2 = ax.bar(x + width/2, cv_rmsle_scores, width, label='CV_RMSLE')

# Add labels and title
ax.set_xlabel('Algorithm')
ax.set_ylabel('Score')
ax.set_title('Comparison of Regression Models (RMSLE vs CV_RMSLE)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Function to add labels to the bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels to the bars
autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```


Comparison of Regression Models (RMSLE vs CV_RMSLE)
RMSLE
CV_RMSLE
0.66
0.46
0.67
0.79
0.72
0.75
0.56
Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
GBR
MLP
NN
XGB
ADA
Algorithm

In [62]:

```
# Data for the bar graph
labels = ['GBR', 'MLP', 'NN', 'XGB', 'ADA']
mae_scores = [round(gbr_mae2, 2), round(mlp_mae2, 2), round(nn_mae2, 2), round(xgb_mae2,
cv_mae_scores = [round(-gbr_cv_mae2, 2), round(-mlp_cv_mae2, 2), round(-nn_cv_mae2, 2),

# Set up the bar graph
x = np.arange(len(labels))
width = 0.35
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, mae_scores, width, label='MAE')
rects2 = ax.bar(x + width/2, cv_mae_scores, width, label='CV_MAE')

# Add labels and title
ax.set_xlabel('Algorithm')
ax.set_ylabel('Score')
ax.set_title('Comparison of Regression Models (MAE vs CV_MAE)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Function to add labels to the bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels to the bars
autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```



Deployment

In [63]:

```
# feature 2 X and y
y = df_clean['Delay'] #store target variable
X = df_clean[['School_Year','Run_Type','Reason','Boro','Bus_Company_Name','Number_Of_Stu
             'School_Age_or_PreK']]

dummy_df = pd.get_dummies(X)

X_train, X_test,y_train, y_test = train_test_split(dummy_df,y,test_size = .2, random_sta
```

In [64]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_log_error
from sklearn.model_selection import GridSearchCV

# Instantiate a new GradientBoostingRegressor object with the best hyperparameters
gbr_best = GradientBoostingRegressor(n_estimators=100, max_depth=4, learning_rate=0.1, l

# Fit the model on the training data
gbr_best.fit(X_train, y_train)

joblib.dump(gbr_best, 'trained_model.pkl')
```

Out[64]:

```
['trained_model.pkl']
```

In [65]:

```
from sklearn.preprocessing import StandardScaler

#row num
row_num = 34

# Best trained model
model = joblib.load('trained_model.pkl')

# input
sample_df = dummy_df.iloc[row_num]
print('actual result:', y[row_num])

# reshape
sample_df = sample_df.values.reshape(1, -1)

# Perform prediction using the loaded model
prediction = model.predict(sample_df)

# Print the prediction result
print('Predicted result:', prediction)
```

```
actual result: 25
Predicted result: [25.00007299]
```