*# importing the necessary libraries*

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** sklearn **as** sk

In [2]:

**import** os**,** types

**import** pandas **as** pd

**from** botocore.client **import** Config

**import** ibm\_boto3

**def** \_\_iter\_\_(self): **return** 0

*# @hidden\_cell*

*# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.*

*# You might want to remove those credentials before you share the notebook.*

client\_b7093f183f79408cb1557752da301291 **=** ibm\_boto3**.**client(service\_name**=**'s3',

ibm\_api\_key\_id**=**'Naw473gQu9CyYhQqbF1gnLY6p9GBU3P0xeqaHhrbFPmF',

ibm\_auth\_endpoint**=**"https://iam.cloud.ibm.com/oidc/token",

config**=**Config(signature\_version**=**'oauth'),

endpoint\_url**=**'https://s3.private.us.cloud-object-storage.appdomain.cloud')

body **=** client\_b7093f183f79408cb1557752da301291**.**get\_object(Bucket**=**'trafficvolumeestimationusingibmwa-donotdelete-pr-1m4tf2iadfc1y8',Key**=**'traffic volume (2).csv')['Body']

*# add missing \_\_iter\_\_ method, so pandas accepts body as file-like object*

**if** **not** hasattr(body, "\_\_iter\_\_"): body**.**\_\_iter\_\_ **=** types**.**MethodType( \_\_iter\_\_, body )

data**=** pd**.**read\_csv(body)

data**.**head()

Out[2]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **date** | **Time** | **traffic\_volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | None | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 09:00:00 | 5545 |
| **1** | None | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | 4516 |
| **2** | None | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | 4767 |
| **3** | None | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | 5026 |
| **4** | None | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | 4918 |

In [3]:

*# importing the data*

*#data = pd.read\_csv(r"C:\Users\Sneka Dharshini\OneDrive\Documents\Traffic\_volume\_estimation-main\Dataset\traffic volume.csv")*

In [4]:

*# displaying first 5 columns of the data*

data**.**head()

Out[4]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **date** | **Time** | **traffic\_volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | None | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 09:00:00 | 5545 |
| **1** | None | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | 4516 |
| **2** | None | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | 4767 |
| **3** | None | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | 5026 |
| **4** | None | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | 4918 |

In [5]:

*# displaying the dimensions of the data (rows and columns)*

data**.**shape

Out[5]:

(48204, 8)

In [6]:

*# used to display the basic information of the data*

data**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 48204 entries, 0 to 48203

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 holiday 48204 non-null object

1 temp 48151 non-null float64

2 rain 48202 non-null float64

3 snow 48192 non-null float64

4 weather 48155 non-null object

5 date 48204 non-null object

6 Time 48204 non-null object

7 traffic\_volume 48204 non-null int64

dtypes: float64(3), int64(1), object(4)

memory usage: 2.9+ MB

In [7]:

*# used to understand the descriptive analysis of the data*

data**.**describe()

Out[7]:

|  | **temp** | **rain** | **snow** | **traffic\_volume** |
| --- | --- | --- | --- | --- |
| **count** | 48151.000000 | 48202.000000 | 48192.000000 | 48204.000000 |
| **mean** | 281.205351 | 0.334278 | 0.000222 | 3259.818355 |
| **std** | 13.343675 | 44.790062 | 0.008169 | 1986.860670 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 272.160000 | 0.000000 | 0.000000 | 1193.000000 |
| **50%** | 282.460000 | 0.000000 | 0.000000 | 3380.000000 |
| **75%** | 291.810000 | 0.000000 | 0.000000 | 4933.000000 |
| **max** | 310.070000 | 9831.300000 | 0.510000 | 7280.000000 |

In [8]:

*# used to display the null values of the data*

data**.**isnull()**.**sum()

Out[8]:

holiday 0

temp 53

rain 2

snow 12

weather 49

date 0

Time 0

traffic\_volume 0

dtype: int64

In [9]:

*# used to display the data type of each column*

data**.**dtypes

Out[9]:

holiday object

temp float64

rain float64

snow float64

weather object

date object

Time object

traffic\_volume int64

dtype: object

**Dealing with missing values of the data**

In [10]:

**from** collections **import** Counter

In [11]:

print(Counter(data['rain']))

print(Counter(data['snow']))

Counter({0.0: 44735, 0.25: 948, 0.51: 256, 1.02: 123, 0.3: 121, 0.76: 109, 0.38: 99, 1.78: 91, 1.52: 69, 0.64: 55, 1.27: 50, 0.6: 32, 2.79: 29, 0.44: 26, 0.89: 25, 2.54: 23, 0.28: 23, 0.42: 21, 1.4: 21, 0.34: 20, 2.16: 19, 2.29: 19, 2.03: 19, 1.8: 16, 1.09: 16, 3.05: 15, 0.32: 15, 1.2: 15, 0.9: 15, 0.98: 14, 0.68: 13, 0.81: 13, 4.57: 13, 7.11: 12, 0.85: 12, 0.7: 11, 2.1: 11, 0.55: 11, 5.59: 10, 1.86: 10, 8.4: 10, 1.15: 10, 0.47: 9, 5.08: 9, 1.21: 9, 0.43: 9, 6.1: 9, 5.84: 8, 1.66: 8, 0.79: 8, 0.4: 8, 1.14: 8, 2.2: 8, 1.85: 8, 2.41: 8, 3.3: 8, 1.41: 7, 6.6: 7, 0.35: 7, 1.91: 7, 0.52: 7, 1.3: 7, 0.8: 7, 0.66: 7, 2.67: 7, 1.33: 7, 1.1: 7, 4.06: 7, 0.57: 6, 0.29: 6, 0.36: 6, 1.0: 6, 1.44: 6, 8.64: 6, 1.35: 6, 5.97: 6, 0.56: 6, 0.91: 6, 0.54: 6, 0.94: 6, 0.96: 6, 2.86: 6, 0.78: 6, 2.22: 6, 6.35: 6, 4.89: 6, 0.93: 6, 3.13: 6, 0.63: 6, 2.62: 6, 1.6: 6, 4.74: 6, 2.76: 6, 3.45: 6, 0.69: 5, 3.18: 5, 0.61: 5, 5.42: 5, 1.39: 5, 0.53: 5, 0.48: 5, 0.59: 5, 0.71: 5, 0.27: 5, 2.85: 5, 0.65: 5, 1.5: 5, 3.41: 5, 4.29: 5, 1.72: 5, 2.61: 5, 1.69: 5, 4.15: 5, 9.62: 5, 0.84: 4, 6.94: 4, 4.32: 4, 1.68: 4, 0.41: 4, 1.06: 4, 2.05: 4, 0.88: 4, 4.45: 4, 5.46: 4, 2.7: 4, 4.21: 4, 9.9: 4, 0.86: 4, 5.92: 4, 10.67: 4, 13.46: 4, 3.94: 4, 20.07: 4, 3.27: 4, 4.0: 4, 2.92: 4, 10.6: 4, 1.34: 3, 1.84: 3, 1.7: 3, 5.74: 3, 4.98: 3, 3.65: 3, 12.19: 3, 7.54: 3, 16.38: 3, 1.65: 3, 3.81: 3, 7.37: 3, 10.54: 3, 19.9: 3, 25.32: 3, 21.42: 3, 9.53: 3, 13.21: 3, 2.37: 3, 3.98: 3, 4.27: 3, 1.13: 3, 0.97: 3, 14.73: 3, 0.95: 3, 1.07: 3, 1.11: 3, 1.24: 3, 3.19: 3, 4.76: 3, 5.27: 3, 11.58: 3, 7.02: 3, 3.08: 3, 1.98: 3, 1.04: 3, 1.55: 3, 6.89: 3, 3.9: 3, 5.02: 3, 4.09: 3, 1.19: 3, 4.8: 3, 4.18: 3, 1.49: 3, 9.4: 3, 3.2: 3, 7.97: 3, 23.8: 3, 11.78: 3, 7.51: 3, 2.15: 3, 9.91: 3, 27.57: 3, 7.29: 3, 13.64: 3, 7.25: 3, 2.91: 3, 20.24: 3, 13.32: 3, 4.38: 3, 3.54: 3, 6.47: 3, 1.56: 3, 8.04: 3, 25.46: 3, 3.74: 3, 2.49: 3, 5.04: 3, 5.36: 3, 2.38: 3, 3.28: 3, 4.04: 3, 3.86: 3, 5.69: 3, 6.01: 3, 5.21: 3, 4.7: 3, 10.92: 3, 7.62: 3, 11.23: 3, 9.42: 3, 10.16: 3, 9.15: 3, 3.75: 3, 1.82: 3, 5.62: 3, 3.1: 3, 2.6: 3, 1.45: 3, 2.26: 3, 2.48: 3, 5.12: 3, 1.01: 3, 4.79: 3, 5.19: 3, 3.39: 2, 0.46: 2, 9.14: 2, 3.56: 2, 16.0: 2, 1.96: 2, 4.39: 2, 28.7: 2, 0.83: 2, 0.72: 2, 0.31: 2, 0.26: 2, 0.58: 2, 1.08: 2, 1.71: 2, 2.21: 2, 0.62: 2, 1.12: 2, 1.46: 2, 1.32: 2, 0.87: 2, 7.39: 2, 1.83: 2, 0.5: 2, 0.77: 2, 2.98: 2, 4.43: 2, 6.45: 2, 1.76: 2, 7.77: 2, 8.89: 2, 15.41: 2, 5.25: 2, 8.02: 2, 12.7: 2, 1.03: 2, 5.86: 2, 7.87: 2, 1.67: 2, 3.09: 2, 1.51: 2, 7.72: 2, 4.64: 2, 2.06: 2, 2.4: 2, 2.96: 2, 6.48: 2, 5.89: 2, 2.39: 2, 2.88: 2, 3.4: 2, 1.29: 1, 44.45: 1, 55.63: 1, 18.8: 1, 0.37: 1, 0.67: 1, 1.87: 1, 0.33: 1, 2.13: 1, 1.63: 1, 1.38: 1, 2.35: 1, 2.11: 1, 2.53: 1, 0.92: 1, 1.22: 1, 1.05: 1, 2.31: 1, 3.17: 1, 2.14: 1, 2.34: 1, 1.61: 1, 5.58: 1, 5.11: 1, 5.1: 1, 4.53: 1, 1.25: 1, 4.5: 1, 3.47: 1, 0.45: 1, 2.18: 1, 2.84: 1, 2.93: 1, 2.87: 1, 2.8: 1, 0.74: 1, 1.28: 1, 1.47: 1, 4.66: 1, 2.08: 1, 3.12: 1, 1.53: 1, 3.25: 1, 1.9: 1, 12.45: 1, 1.37: 1, 2.78: 1, 1.31: 1, 3.44: 1, 2.75: 1, 2.19: 1, 1.59: 1, 5.73: 1, 5.93: 1, 3.91: 1, 18.03: 1, 1.88: 1, 3.01: 1, 2.12: 1, 0.73: 1, 11.59: 1, 2.33: 1, 5.52: 1, 1.93: 1, 2.68: 1, 10.05: 1, 7.7: 1, 4.05: 1, 3.8: 1, 9831.3: 1, 16.51: 1, 12.83: 1, 18.42: 1, 5.06: 1, 1.95: 1, 9.0: 1, 8.86: 1, 5.99: 1, 8.0: 1, 31.75: 1, 5.41: 1, 2.83: 1, 15.75: 1, 3.64: 1, 7.13: 1, 1.16: 1, 7.05: 1, 2.73: 1, nan: 1, nan: 1})

Counter({0.0: 48129, 0.05: 14, 0.06: 12, 0.51: 6, 0.25: 6, 0.13: 6, 0.1: 6, 0.32: 5, 0.17: 3, 0.44: 2, 0.08: 2, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 0.21: 1, nan: 1})

In [12]:

data['temp']**.**fillna(data['temp']**.**mean(),inplace**=True**)

data['rain']**.**fillna(data['rain']**.**mean(),inplace**=True**)

data['snow']**.**fillna(data['snow']**.**mean(),inplace**=True**)

In [13]:

print(Counter(data['weather']))

Counter({'Clouds': 15144, 'Clear': 13383, 'Mist': 5942, 'Rain': 5665, 'Snow': 2875, 'Drizzle': 1818, 'Haze': 1359, 'Thunderstorm': 1033, 'Fog': 912, nan: 49, 'Smoke': 20, 'Squall': 4})

In [14]:

data['weather']**.**fillna('Clouds',inplace**=True**)

In [15]:

data**.**isnull()**.**sum()

Out[15]:

holiday 0

temp 0

rain 0

snow 0

weather 0

date 0

Time 0

traffic\_volume 0

dtype: int64

**encoding the data**

In [16]:

**from** sklearn.preprocessing **import** LabelEncoder

In [17]:

le **=** LabelEncoder()

In [18]:

data['weather'] **=** le**.**fit\_transform(data['weather'])

In [19]:

data['holiday'] **=** le**.**fit\_transform(data['holiday'])

In [20]:

data**.**head()

Out[20]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **date** | **Time** | **traffic\_volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7 | 288.28 | 0.0 | 0.0 | 1 | 02-10-2012 | 09:00:00 | 5545 |
| **1** | 7 | 289.36 | 0.0 | 0.0 | 1 | 02-10-2012 | 10:00:00 | 4516 |
| **2** | 7 | 289.58 | 0.0 | 0.0 | 1 | 02-10-2012 | 11:00:00 | 4767 |
| **3** | 7 | 290.13 | 0.0 | 0.0 | 1 | 02-10-2012 | 12:00:00 | 5026 |
| **4** | 7 | 291.14 | 0.0 | 0.0 | 1 | 02-10-2012 | 13:00:00 | 4918 |

In [21]:

*# spliiting the date column into year,month,day*

data[["day", "month", "year"]] **=** data["date"]**.**str**.**split("-", expand **=** **True**)

In [22]:

*# spliiting the date column into year,month,day*

data[["hours", "minutes", "seconds"]] **=** data["Time"]**.**str**.**split(":", expand **=** **True**)

In [23]:

data**.**drop(columns**=**['date','Time'],axis**=**1,inplace**=True**)

In [24]:

y **=** data['traffic\_volume']

x **=** data**.**drop(columns**=**['traffic\_volume'],axis**=**1)

In [25]:

data**.**head()

Out[25]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **traffic\_volume** | **day** | **month** | **year** | **hours** | **minutes** | **seconds** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7 | 288.28 | 0.0 | 0.0 | 1 | 5545 | 02 | 10 | 2012 | 09 | 00 | 00 |
| **1** | 7 | 289.36 | 0.0 | 0.0 | 1 | 4516 | 02 | 10 | 2012 | 10 | 00 | 00 |
| **2** | 7 | 289.58 | 0.0 | 0.0 | 1 | 4767 | 02 | 10 | 2012 | 11 | 00 | 00 |
| **3** | 7 | 290.13 | 0.0 | 0.0 | 1 | 5026 | 02 | 10 | 2012 | 12 | 00 | 00 |
| **4** | 7 | 291.14 | 0.0 | 0.0 | 1 | 4918 | 02 | 10 | 2012 | 13 | 00 | 00 |

**scaling the data**

In [26]:

y **=** data['traffic\_volume']

x **=** data**.**drop(columns**=**['traffic\_volume'],axis**=**1)

In [27]:

names **=** x**.**columns

In [28]:

**from** sklearn.preprocessing **import** scale

In [29]:

x **=** scale(x)

In [30]:

x **=** pd**.**DataFrame(x,columns**=**names)

In [31]:

x**.**head()

Out[31]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **day** | **month** | **year** | **hours** | **minutes** | **seconds** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.015856 | 0.530485 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.345548 | 0.0 | 0.0 |
| **1** | 0.015856 | 0.611467 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.201459 | 0.0 | 0.0 |
| **2** | 0.015856 | 0.627964 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.057371 | 0.0 | 0.0 |
| **3** | 0.015856 | 0.669205 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | 0.086718 | 0.0 | 0.0 |
| **4** | 0.015856 | 0.744939 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | 0.230807 | 0.0 | 0.0 |

**splitting the data**

In [32]:

**from** sklearn.model\_selection **import** train\_test\_split

In [33]:

x\_train,x\_test,y\_train,y\_test **=** train\_test\_split(x,y,test\_size**=**0.2,random\_state**=**0)

**Model building**

In [34]:

**from** sklearn **import** linear\_model

**from** sklearn **import** tree

**from** sklearn **import** ensemble

**from** sklearn **import** svm

**import** xgboost

In [35]:

lin\_reg **=** linear\_model**.**LinearRegression()

Dtree **=** tree**.**DecisionTreeRegressor()

Rand **=** ensemble**.**RandomForestRegressor()

svr **=** svm**.**SVR()

XGB **=** xgboost**.**XGBRegressor()

In [36]:

lin\_reg**.**fit(x\_train,y\_train)

Dtree**.**fit(x\_train,y\_train)

Rand**.**fit(x\_train,y\_train)

svr**.**fit(x\_train,y\_train)

XGB**.**fit(x\_train,y\_train)

Out[36]:

XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bynode=1, colsample\_bytree=1, enable\_categorical=False,

gamma=0, gpu\_id=-1, importance\_type=None,

interaction\_constraints='', learning\_rate=0.300000012,

max\_delta\_step=0, max\_depth=6, min\_child\_weight=1, missing=nan,

monotone\_constraints='()', n\_estimators=100, n\_jobs=56,

num\_parallel\_tree=1, predictor='auto', random\_state=0, reg\_alpha=0,

reg\_lambda=1, scale\_pos\_weight=1, subsample=1, tree\_method='exact',

validate\_parameters=1, verbosity=None)

In [37]:

p1 **=** lin\_reg**.**predict(x\_train)

p2 **=** Dtree**.**predict(x\_train)

p3 **=** Rand**.**predict(x\_train)

p4 **=** svr**.**predict(x\_train)

p5 **=** XGB**.**predict(x\_train)

In [38]:

**from** sklearn **import** metrics

In [39]:

print(metrics**.**r2\_score(p1,y\_train))

print(metrics**.**r2\_score(p2,y\_train))

print(metrics**.**r2\_score(p3,y\_train))

print(metrics**.**r2\_score(p4,y\_train))

print(metrics**.**r2\_score(p5,y\_train))

-5.517285423636859

1.0

0.9747734895676703

-12.188104231382287

0.8349874938269883

**with testing data finding the r-score**

In [40]:

p1 **=** lin\_reg**.**predict(x\_test)

p2 **=** Dtree**.**predict(x\_test)

p3 **=** Rand**.**predict(x\_test)

p4 **=** svr**.**predict(x\_test)

p5 **=** XGB**.**predict(x\_test)

In [41]:

print(metrics**.**r2\_score(p1,y\_test))

print(metrics**.**r2\_score(p2,y\_test))

print(metrics**.**r2\_score(p3,y\_test))

print(metrics**.**r2\_score(p4,y\_test))

print(metrics**.**r2\_score(p5,y\_test))

-5.399396398322177

0.6931650434117814

0.8032230428060344

-11.972215715232434

0.7922184852381723

**Randforest gives the best r-score value**

In [42]:

*#RMSE values*

MSE **=** metrics**.**mean\_squared\_error(p3,y\_test)

In [43]:

np**.**sqrt(MSE)

Out[43]:

798.4391472199028

**saving the model**

In [44]:

**import** pickle

In [45]:

pickle**.**dump(Rand,open("model.pkl",'wb'))

pickle**.**dump(le,open("encoder.pkl",'wb'))

In [46]:

data**.**head()

x**.**head()

Out[46]:

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **day** | **month** | **year** | **hours** | **minutes** | **seconds** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.015856 | 0.530485 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.345548 | 0.0 | 0.0 |
| **1** | 0.015856 | 0.611467 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.201459 | 0.0 | 0.0 |
| **2** | 0.015856 | 0.627964 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | -0.057371 | 0.0 | 0.0 |
| **3** | 0.015856 | 0.669205 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | 0.086718 | 0.0 | 0.0 |
| **4** | 0.015856 | 0.744939 | -0.007463 | -0.027235 | -0.566452 | -1.574903 | 1.02758 | -1.855294 | 0.230807 | 0.0 | 0.0 |

In [47]:

y**.**head()

Out[47]:

0 5545

1 4516

2 4767

3 5026

4 4918

Name: traffic\_volume, dtype: int64

In [48]:

x**.**shape

Out[48]:

(48204, 11)

In [49]:

y**.**shape

Out[49]:

(48204,)

In [50]:

**!**pip install ibm\_watson\_machine\_learning

Requirement already satisfied: ibm\_watson\_machine\_learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.238)

Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (2022.6.15)

Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (0.8.9)

Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (4.8.2)

Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (1.26.7)

Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (2.26.0)

Requirement already satisfied: ibm-cos-sdk==2.11.\* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (2.11.0)

Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (21.3)

Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (0.3.3)

Requirement already satisfied: pandas<1.4.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm\_watson\_machine\_learning) (1.3.4)

Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.\*->ibm\_watson\_machine\_learning) (0.10.0)

Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.\*->ibm\_watson\_machine\_learning) (2.11.0)

Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.\*->ibm\_watson\_machine\_learning) (2.11.0)

Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.\*->ibm\_watson\_machine\_learning) (2.8.2)

Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.4.0,>=0.24.2->ibm\_watson\_machine\_learning) (2021.3)

Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.4.0,>=0.24.2->ibm\_watson\_machine\_learning) (1.20.3)

Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.\*->ibm\_watson\_machine\_learning) (1.15.0)

Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm\_watson\_machine\_learning) (3.3)

Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm\_watson\_machine\_learning) (2.0.4)

Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm\_watson\_machine\_learning) (3.6.0)

Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm\_watson\_machine\_learning) (3.0.4)

In [51]:

**from** ibm\_watson\_machine\_learning **import** APIClient

wml\_credentials **=** {

"url": "https://us-south.ml.cloud.ibm.com",

"apikey":"WSUwzDwBwUJSv5sUOaQ9Hg8foR8dIPLvfidK0k\_4ylxE"

}

client **=** APIClient(wml\_credentials)

In [52]:

**def** guid\_from\_space\_name(client, space\_name):

space **=** client**.**spaces**.**get\_details()

*#print (space)*

**return**(next(item **for** item **in** space['resources'] **if** item['entity']["name"] **==** space\_name)['metadata']['id'])

In [53]:

space\_uid **=** guid\_from\_space\_name(client, "Newspace")

print("Space UID = " **+** space\_uid)

Space UID = 1dcf6048-00c2-45cb-ad8a-0b46044df22c

In [54]:

client**.**set**.**default\_space(space\_uid)

Out[54]:

'SUCCESS'

In [55]:

client**.**software\_specifications**.**list()

----------------------------- ------------------------------------ ----

NAME ASSET\_ID TYPE

default\_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base

kernel-spark3.2-scala2.12 020d69ce-7ac1-5e68-ac1a-31189867356a base

pytorch-onnx\_1.3-py3.7-edt 069ea134-3346-5748-b513-49120e15d288 base

scikit-learn\_0.20-py3.6 09c5a1d0-9c1e-4473-a344-eb7b665ff687 base

spark-mllib\_3.0-scala\_2.12 09f4cff0-90a7-5899-b9ed-1ef348aebdee base

pytorch-onnx\_rt22.1-py3.9 0b848dd4-e681-5599-be41-b5f6fccc6471 base

ai-function\_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda base

shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base

tensorflow\_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22 base

pytorch\_1.1-py3.6 10ac12d6-6b30-4ccd-8392-3e922c096a92 base

tensorflow\_1.15-py3.6-ddl 111e41b3-de2d-5422-a4d6-bf776828c4b7 base

runtime-22.1-py3.9 12b83a17-24d8-5082-900f-0ab31fbfd3cb base

scikit-learn\_0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base

default\_r3.6 1b70aec3-ab34-4b87-8aa0-a4a3c8296a36 base

pytorch-onnx\_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base

pytorch-onnx\_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f base

tensorflow\_2.1-py3.6 1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base

spark-mllib\_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base

tensorflow\_2.4-py3.8-horovod 217c16f6-178f-56bf-824a-b19f20564c49 base

runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce658 base

do\_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base

autoai-ts\_3.8-py3.8 2aa0c932-798f-5ae9-abd6-15e0c2402fb5 base

tensorflow\_1.15-py3.6 2b73a275-7cbf-420b-a912-eae7f436e0bc base

pytorch\_1.2-py3.6 2c8ef57d-2687-4b7d-acce-01f94976dac1 base

spark-mllib\_2.3 2e51f700-bca0-4b0d-88dc-5c6791338875 base

pytorch-onnx\_1.1-py3.6-edt 32983cea-3f32-4400-8965-dde874a8d67e base

spark-mllib\_3.0-py37 36507ebe-8770-55ba-ab2a-eafe787600e9 base

spark-mllib\_2.4 390d21f8-e58b-4fac-9c55-d7ceda621326 base

xgboost\_0.82-py3.6 39e31acd-5f30-41dc-ae44-60233c80306e base

pytorch-onnx\_1.2-py3.6-edt 40589d0e-7019-4e28-8daa-fb03b6f4fe12 base

default\_r36py38 41c247d3-45f8-5a71-b065-8580229facf0 base

autoai-ts\_rt22.1-py3.9 4269d26e-07ba-5d40-8f66-2d495b0c71f7 base

autoai-obm\_3.0 42b92e18-d9ab-567f-988a-4240ba1ed5f7 base

pmml-3.0\_4.3 493bcb95-16f1-5bc5-bee8-81b8af80e9c7 base

spark-mllib\_2.4-r\_3.6 49403dff-92e9-4c87-a3d7-a42d0021c095 base

xgboost\_0.90-py3.6 4ff8d6c2-1343-4c18-85e1-689c965304d3 base

pytorch-onnx\_1.1-py3.6 50f95b2a-bc16-43bb-bc94-b0bed208c60b base

autoai-ts\_3.9-py3.8 52c57136-80fa-572e-8728-a5e7cbb42cde base

spark-mllib\_2.4-scala\_2.11 55a70f99-7320-4be5-9fb9-9edb5a443af5 base

spark-mllib\_3.0 5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9 base

autoai-obm\_2.0 5c2e37fa-80b8-5e77-840f-d912469614ee base

spss-modeler\_18.1 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base

cuda-py3.8 5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e base

autoai-kb\_3.1-py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base

pytorch-onnx\_1.7-py3.8 634d3cdc-b562-5bf9-a2d4-ea90a478456b base

spark-mllib\_2.3-r\_3.6 6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c base

tensorflow\_2.4-py3.7 65e171d7-72d1-55d9-8ebb-f813d620c9bb base

spss-modeler\_18.2 687eddc9-028a-4117-b9dd-e57b36f1efa5 base

pytorch-onnx\_1.2-py3.6 692a6a4d-2c4d-45ff-a1ed-b167ee55469a base

spark-mllib\_2.3-scala\_2.11 7963efe5-bbec-417e-92cf-0574e21b4e8d base

----------------------------- ------------------------------------ ----

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

In [56]:

software\_spec\_uid **=** client**.**software\_specifications**.**get\_uid\_by\_name("runtime-22.1-py3.9")

software\_spec\_uid

Out[56]:

'12b83a17-24d8-5082-900f-0ab31fbfd3cb'

In [58]:

model\_details **=** client**.**repository**.**store\_model(model**=**Dtree,meta\_props**=**{

client**.**repository**.**ModelMetaNames**.**NAME:"traffic\_modeling",

client**.**repository**.**ModelMetaNames**.**TYPE:"scikit-learn\_1.0",

client**.**repository**.**ModelMetaNames**.**SOFTWARE\_SPEC\_UID:software\_spec\_uid }

)

model\_id **=** client**.**repository**.**get\_model\_uid(model\_details)

This method is deprecated, please use get\_model\_id()

In [59]:

model\_id

Out[59]:

'06cddb46-123d-42f3-ba7b-fefac0f52977'

In [ ]: