21162101007_KSHITIJGUPTA

Data Set-1: Metro Interstate Traffic

```
from google.colab import files
uploaded = files.upload()
```

Choose files Metro_Inter…_Volume.csv

- **Metro_Interstate_Traffic_Volume.csv**(text/csv) - 3237208 bytes, last modified: 08/05/2019 - 100% done

```
Saving Metro_Interstate_Traffic_Volume.csv to Metro_Interstate_Traffic_Volume.csv
```

```
import pandas as pd
import numpy as np
import io
```

```
df = pd.read_csv(io.BytesIO(uploaded['Metro_Interstate_Traffic_Volume.csv']))
print(df.shape)
df.head()
df
```

```
(48204, 9)
```

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description
0	NaN	288.28	0.0	0.0	40	Clouds	scattered clouds
1	NaN	289.36	0.0	0.0	75	Clouds	broken clouds
2	NaN	289.58	0.0	0.0	90	Clouds	overcast clouds
3	NaN	290.13	0.0	0.0	90	Clouds	overcast clouds
4	NaN	291.14	0.0	0.0	75	Clouds	broken clouds
...	...	...	...	...	...	...	...

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
from datetime import datetime
def datetimeConv(value):
  return datetime.strptime(value,'%Y-%m-%d %H:%M:%S').time()

df['time'] = df['date_time'].apply(datetimeConv)


df = df.drop("date_time",axis=1)


df
```

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description
0	NaN	288.28	0.0	0.0	40	Clouds	scattered clouds
1	NaN	289.36	0.0	0.0	75	Clouds	broken clouds
2	NaN	289.58	0.0	0.0	90	Clouds	overcast clouds
3	NaN	290.13	0.0	0.0	90	Clouds	overcast clouds
4	NaN	291.14	0.0	0.0	75	Clouds	broken clouds
...	...	...	...	...	...	...	...
48199	NaN	283.45	0.0	0.0	75	Clouds	broken clouds
48200	NaN	282.76	0.0	0.0	90	Clouds	overcast clouds
48201	NaN	282.73	0.0	0.0	90	Thunderstorm	proximity thunderstorm
48202	NaN	282.09	0.0	0.0	90	Clouds	overcast clouds
48203	NaN	282.12	0.0	0.0	90	Clouds	overcast clouds

48204 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
def holidayConversion(value):
  if 'none' in value.lower():
    return 0
  else:
    return 1

df['holiday'] = df['holiday'].map(str).apply(holidayConversion)
df.head()
```

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traf
0	1	288.28	0.0	0.0	40	Clouds	scattered clouds	
1	1	289.36	0.0	0.0	75	Clouds	broken clouds	
2	1	289.58	0.0	0.0	90	Clouds	overcast clouds	
3	1	290.13	0.0	0.0	90	Clouds	overcast clouds	
4	1	291.14	0.0	0.0	75	Clouds	broken clouds	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()


df['weather_main'] = le.fit_transform(df['weather_main'])
df['weather_description'] = le.fit_transform(df['weather_description'])
df['time'] = le.fit_transform(df['time'])


X = df.drop("traffic_volume",axis=1)
y = df["traffic_volume"]


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor


# Split the data for testing and training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)


#creating a model
model = LinearRegression()
model.fit(X_train, y_train)

#predicting
predictions = model.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False)

print("Linear Regression\n\t Mean Square Error: ",rmse,"\n\t r2 score: ",r2_score(y_test,pre
```

```
Linear Regression
         Mean Square Error:  1859.8652639516467
         r2 score:  0.13250293486622244
```

```
#creating a model
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

#predicting
predictions = model.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False)

print("Random Forest Tree Regression\n\t Mean Square Error: ",rmse,"\n\t r2 score: ",r2_scor
```

```
Random Forest Tree Regression
         Mean Square Error:  948.5000522031985
         r2 score:  0.7743788722046312
```

Choosing Random Forest as a Regression Model as it has minimum MSE = 948.091 and highest r2 score = 0.774

Data: Rental Bike Share Prediction

Set-1: hour

```
from google.colab import files
uploaded = files.upload()
```

Choose files hour.csv

- **hour.csv**(text/csv) - 1156736 bytes, last modified: 20/12/2013 - 100% done

```
Saving hour.csv to hour.csv
```

```
df = pd.read_csv(io.BytesIO(uploaded['hour.csv']))
print(df.shape)
df.head()
df
```

```
(17379, 17)
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	t
0	1	2011-01-01	1	0	1	0	0	6	0	1	
1	2	2011-01-01	1	0	1	1	0	6	0	1	
2	3	2011-01-01	1	0	1	2	0	6	0	1	
3	4	2011-01-01	1	0	1	3	0	6	0	1	
4	5	2011-01-01	1	0	1	4	0	6	0	1	
...	...	...	...	...	...	...	...	...	...	...	
17374	17375	2012-12-31	1	1	12	19	0	1	1	2	
17375	17376	2012-12-31	1	1	12	20	0	1	1	2	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Check for null values
df.isnull().sum()
```

	0
instant	0
dteday	0
season	0
yr	0
mnth	0
hr	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

dtype: int64

```
df.drop(['instant', 'dteday', 'casual', 'registered'], axis=1, inplace=True)
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df.drop('cnt', axis=1)
y = df['cnt']

# Split the data for testing and training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

#creating a model
model = LinearRegression()
model.fit(X_train, y_train)

#predicting
predictions = model.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False)

print("Linear Regression\n\t Mean Square Error: ",rmse,"\n\t r2 score: ",r2_score(y_test,pre
```

```
Linear Regression
             Mean Square Error:  143.09906538490392
             r2 score:  0.3800683930842459
```

```
from sklearn.ensemble import RandomForestRegressor

#creating a model
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

#predicting
predictions = model.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False)

print("Random Forest Tree Regression\n\t Mean Square Error: ",rmse,"\n\t r2 score: ",r2_scor
```

```
Random Forest Tree Regression
             Mean Square Error:  42.628815278563366
             r2 score:  0.9449855581942926
```

Choosing Random Forest as a Regression Model as it has minimum MSE = 42.628 and highest r2 score = 0.944

Data: Rental Bike Share Prediction

Set-2: day

```
from google.colab import files
uploaded = files.upload()
```

Choose files day.csv

- **day.csv**(text/csv) - 57569 bytes, last modified: 20/12/2013 - 100% done

```
Saving day.csv to day.csv
```

```
df = pd.read_csv(io.BytesIO(uploaded['day.csv']))
print(df.shape)
df.head()
df
```

```
(731, 16)
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	tem
0	1	2011-01-01	1	0	1	0	6	0	2	0.34416
1	2	2011-01-02	1	0	1	0	0	0	2	0.36347
2	3	2011-01-03	1	0	1	0	1	1	1	0.19636
3	4	2011-01-04	1	0	1	0	2	1	1	0.20000
4	5	2011-01-05	1	0	1	0	3	1	1	0.22695
...	...	...	...	...	...	...	...	...	...	
726	727	2012-12-27	1	1	12	0	4	1	2	0.25416
727	728	2012-12-28	1	1	12	0	5	1	2	0.25333

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Check for null values
df.isnull().sum()
```

	0
instant	0
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
num	0

```
df.drop(['instant', 'dteday', 'casual', 'registered'], axis=1, inplace=True)
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df.drop('cnt', axis=1)
y = df['cnt']

# Split the data for testing and training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)


from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

#creating a model
model = LinearRegression()
model.fit(X_train, y_train)

#predicting
predictions = model.predict(X_test)
```