assign1.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 6 14:13:13 2022

@author: Admin
"""

# First let's start with calling all the dependencies for this project
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

df = pd.read_csv('G:/dypiemr2/dypiemr22-23/sem_I/BE_I/databse/uber.csv')
df.head()

df.shape

df.info()

#find any null value present
df.isnull().sum()

#drop null rows
df.dropna(axis=0,inplace=True)
df.isnull().sum()

#Calculatin the distance between the pickup and drop co-ordinates
#using the Haversine formual for accuracy.
def haversine (lon_1, lon_2, lat_1, lat_2):

lon_1, lon_2, lat_1, lat_2 = map(np.radians, [lon_1, lon_2, lat_1, lat_2])
#Degrees to Radians


diff_lon = lon_2 - lon_1
diff_lat = lat_2 - lat_1


km = 2 * 6371 * np.arcsin(np.sqrt(np.sin(diff_lat/2.0)**2 +
np.cos(lat_1) * np.cos(lat_2) * np.sin(diff_lon/2.0)**2))

return km


#find distance travelled per ride
df['Distance']= haversine(df['pickup_longitude'],df['dropoff_longitude'],
df['pickup_latitude'],df['dropoff_latitude'])
```

```
#round it to 2 decimal points
df['Distance'] = df['Distance'].astype(float).round(2)
df.head()

plt.scatter(df['Distance'], df['fare_amount'])
plt.xlabel("Distance")
plt.ylabel("fare_amount")

#Outliers
#We can get rid of the trips with very large distances that are outliers
# as well as trips with 0 distance.
df.drop(df[df['Distance'] > 60].index, inplace = True)
df.drop(df[df['Distance'] == 0].index, inplace = True)
df.drop(df[df['fare_amount'] == 0].index, inplace = True)
df.drop(df[df['fare_amount'] < 0].index, inplace = True)
df.shape

# removing rows with non-plausible fare amounts and distance travelled
df.drop(df[(df['fare_amount']>100) & (df['Distance']<1)].index, inplace = True
)
df.drop(df[(df['fare_amount']<100) & (df['Distance']>100)].index, inplace =
True )
df.shape

plt.scatter(df['Distance'], df['fare_amount'])
plt.xlabel("Distance")
plt.ylabel("fare_amount")

df.info()

# Create New DataFrame of Specific column
df2 = pd.DataFrame().assign(fare=df['fare_amount'], Distance=df['Distance'])
df2.info()

df2.shape


# plot target fare distribution
plt.figure(figsize=[8,4])
sns.distplot(df2['fare'], color='g',hist_kws=dict(edgecolor="black",
linewidth=2), bins=30)
plt.title('Target Variable Distribution')
plt.show()

#plots
plt.scatter(df2['Distance'], df2['fare'])
plt.xlabel("Distance")
plt.ylabel("fare_amount")


x=df2['fare']
y=df2['Distance']

#independant variable
X = df2['Distance'].values.reshape(-1, 1)
```

```
#dependant variable
Y= df2['fare'].values.reshape(-1, 1)

# scale by standardscalar
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
y_std = std.fit_transform(Y)
x_std = std.fit_transform(X)


#split in test-train
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_std, y_std,
test_size=0.2, random_state=0)

#simple linear regression
from sklearn.linear_model import LinearRegression
l_reg = LinearRegression()
l_reg.fit(X_train, y_train)

#predict test values
y_pred = l_reg.predict(X_test)



#find the error
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
y_pred)))

#final plot
plt.subplot(2, 2, 1)
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.title("Fare vs Distance (Training Set)")
plt.ylabel("fare_amount")
plt.xlabel("Distance")

plt.subplot(2, 2, 2)
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.ylabel("fare_amount")
plt.xlabel("Distance")
plt.title("Fare vs Distance (Test Set)")
```