PART 1

Write the code to import pandas library in Python

Answer: import pandas as pd

Question 2 Not yet answered Marked out of 1.00 Flag question

Write the code to load the dataset 'dsExam1.csv' and store to 'df1' dataframe

Answer: df1 = pd.read_csv('dsExam1.csv')

Question 3 Not yet answered Marked out of 1.00 Flag question

Write the code to Load the dataset 'dsExam4.csv' and store to 'df2' data frame

Answer: df2 = pd.read_csv('dsExam4.csv')

Question 4 Not yet answered Marked out of 1.00 Flag question

```
Perform the following prediction tasks in python.
```

Take a peek at the first 10 entries of '**df1**' data frame. Assume that **df1** contains some data.

Answer: print(df1.head(10))

df1.head(10)

Question 5 Not yet answered Marked out of 1.00 Flag question

```
Perform the following prediction tasks in python.
```

Identify the number of rows and columns of the data frame '**df2**'. Assume that **df2** contains some data.

Answer: row = len(df2) col = len(df2.columns)

df2.shape

```
In [5]:
import pandas as pd
df1 = pd.read_csv('dsExam1.csv')
df2 = pd.read_csv('dsExam4.csv')
df1.head(10)
```

Out[5]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	fuel_encoded	transmission_encoded	owner_encoded	seller_type_encoded
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner	4	1	0	1
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner	4	1	0	1
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner	1	1	0	1
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner	4	1	0	1
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner	1	1	2	1
5	Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner	4	1	0	1
6	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner	4	1	0	1
7	Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner	4	1	2	1
8	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner	4	1	0	1
9	Maruti Celerio Green VXI	2017	365000	78000	CNG	Individual	Manual	First Owner	0	1	0	1

```
In [15]:
df2.shape
```

```
Out[15]: (1000, 20)
```

PART 2

is use when the variable to be predicted is categorical

The dataset is best suited for regression analysis its because

Question 1 Not yet answered Marked out of 5.00 Flag question

Take a look at **'df1'** data frame, discuss why the dataset is best suited for regression analysis. Name or list down the the possible input and target variable.

Why the dataset is best suited for regression analysis(Linear) because the target variable is categorical.
The possible input variable (name, year, selling_price, km_driven, fuel, seller_type, transmission, owner, fuel_encoded, transmission_encoded, owner_encoded, seller_type_encoded) and the possible target/output vairable is year encoded

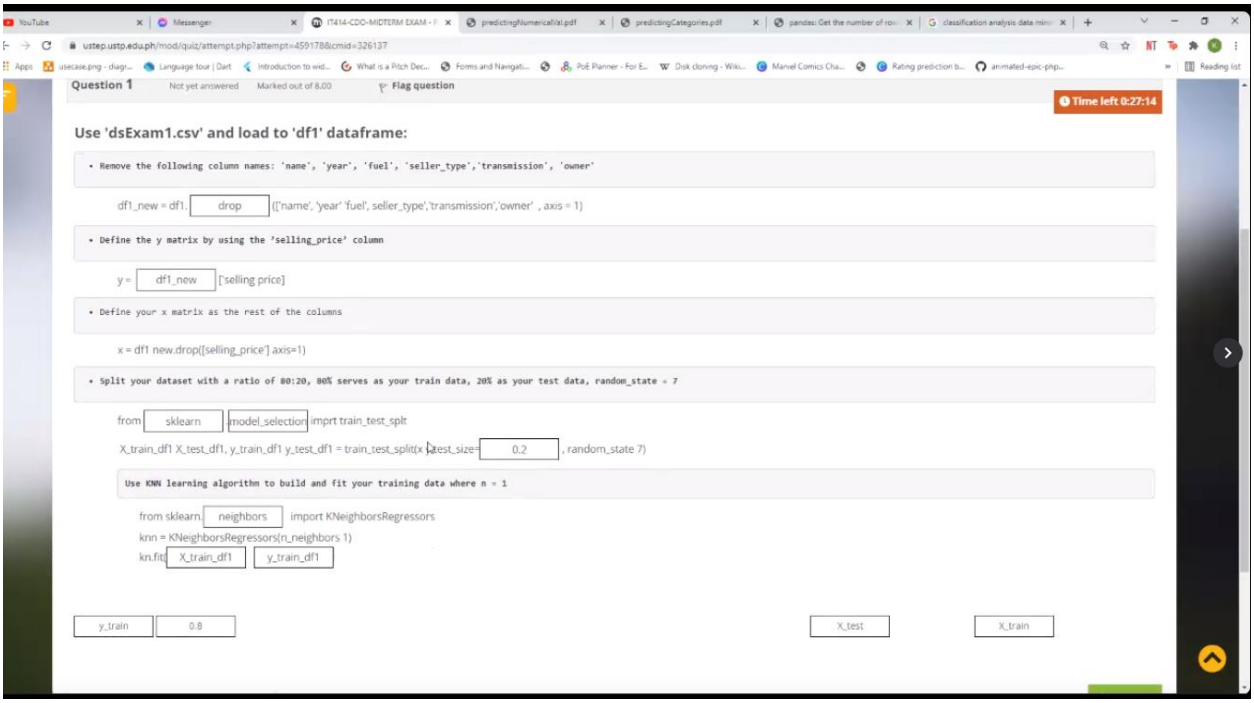
Question 2 Not yet answered Marked out of 5.00 Flag question Time left 0:02:03

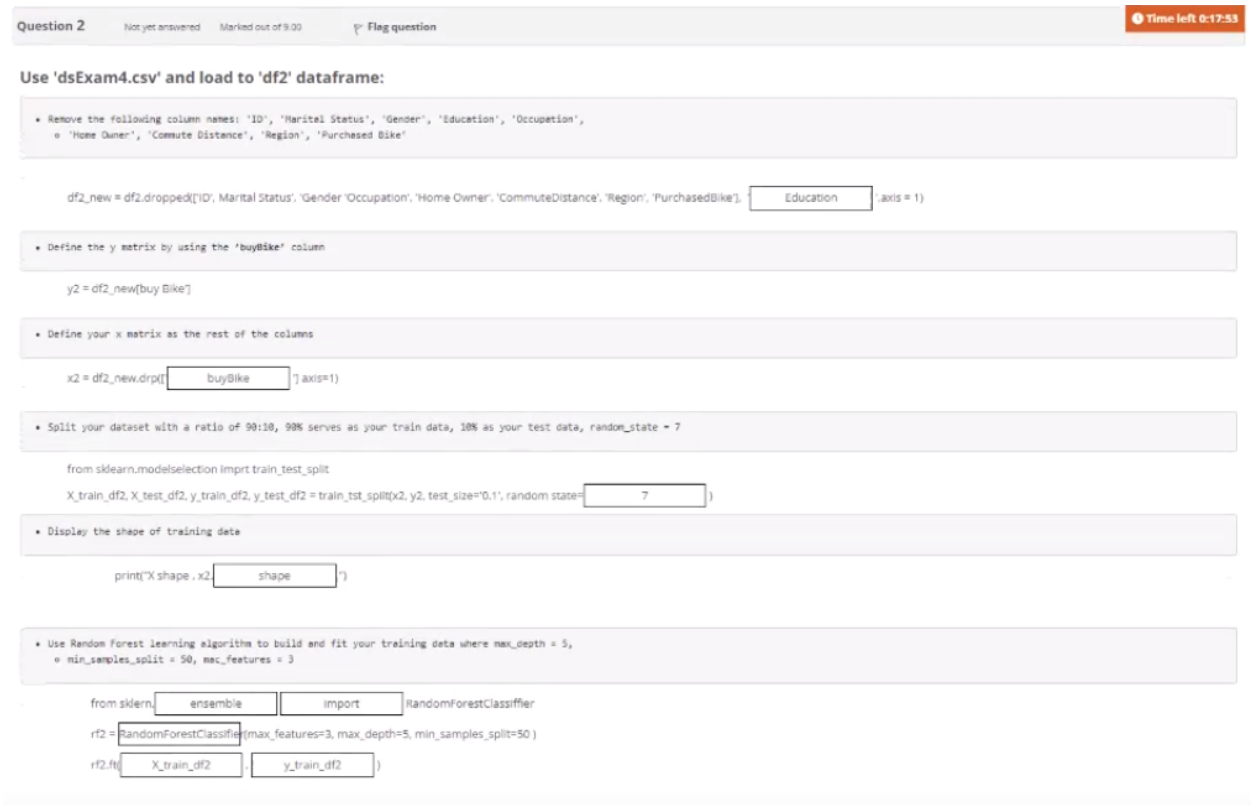
Take a look at '**df2**' data frame, discuss why the dataset is best suited for classification analysis. Name or list down the possible input and target variable.

Looking at 'df2' data frame, the dataset is best suited for classification analysis are used when the variable to be predicted is a categorical.

The possible input variable is (ID, Marital Status, Gender, Income, Children, Education, Occupation, Home Owner, Cars, Commute Distance, Region, Age, Purchased Bike, MS_encoded, gender_encoded, educ_encoded, Occupation_encoded, HO_encoded, CD_encoded) and the possible output or target variable is buyBike.

PART 3


Question 1
Not yet answered
Marked out of 8.00
Flag question
Time left 0:27:14
Use 'dsExam1.csv' and load to 'df1' dataframe:
• Remove the following column names: 'name', 'year', 'fuel', 'seller_type','transmission', 'owner'
df1_new = df1. drop (['name', 'year' 'fuel', seller_type','transmission','owner' , axis = 1)
• Define the y matrix by using the 'selling_price' column
y = df1_new ['selling price]
• Define your x matrix as the rest of the columns
x = df1 new.drop(['selling_price'] axis=1)
• Split your dataset with a ratio of 80:20, 80% serves as your train data, 20% as your test data, random_state = 7
from sklearn model_selection imprt train_test_splt
X_train_df1 X_test_df1, y_train_df1 y_test_df1 = train_test_split(x , test_size= 0.2 , random_state 7)
Use KNN learning algorithm to build and fit your training data where n = 1
from sklearn. neighbors import KNeighborsRegressors
knn = KNeighborsRegressors(n_neighbors 1)
kn.fit(X_train_df1 y_train_df1
y_train
0.8
X_test
X_train


Question 2
Not yet answered
Marked out of 9.00
Flag question
Time left 0:17:53
Use 'dsExam4.csv' and load to 'df2' dataframe:
• Remove the following column names: 'ID', 'Marital Status', 'Gender', 'Education', 'Occupation',
'Home Owner', 'Commute Distance', 'Region', 'Purchased Bike'
df2_new = df2.dropped['ID', Marital Status', 'Gender 'Occupation', 'Home Owner', 'CommuteDistance', 'Region', 'PurchasedBike'], Education ',axis = 1)
• Define the y matrix by using the 'buyBike' column
y2 = df2_new[buy Bike']
• Define your x matrix as the rest of the columns
x2 = df2_new.drp[' buyBike '] axis=1)
• Split your dataset with a ratio of 90:10, 90% serves as your train data, 10% as your test data, random_state = 7
from sklearn.modelselection imprt train_test_split
X_train_df2, X_test_df2, y_train_df2, y_test_df2 = train_tst_split(x2, y2, test_size='0.1', random state= 7)
• Display the shape of training data
print("X shape , x2. shape ")
• Use Random Forest learning algorithm to build and fit your training data where max_depth = 5,
min_samples_split = 50, mac_features = 3
from sklern. ensemble import RandomForestClassifier
rf2 = RandomForestClassifier(max_features=3, max_depth=5, min_samples_split=50)
rf2.ft(X_train_df2 , y_train_df2)

```
Perform Prediction for training data --- rf2 is the name of the mod
```

```
y_pred_rf = rf2.predict(|                |)
accuracy_rf = accuracy_scores(y_true=y_trains_df2, y_pred=                ))
print(accuracy_rf))
```

Output:

```
Performing Prediction for test data
```

```
y_pred_rf2 = rf2.predict(                )
accuracy_rf2 = accuracy_scores(y_true=                , y_preds=y_pred_rf2
print(accuracy_rf2)
```

output:

PART 4

```
Performing Prediction for train data
```

```
y_pred_knn = knn.predict(  X_train_df1  ))
mse_knn = mean_square_error((y_true=  y_train_df1  , y_pred=y_pred_knn)
print(mse_knn/1e6)))
```

Output: 113641.903

```
Performing Prediction for test data
```

```
y_pred_knn2 = kn.predict(  X_test_df1  )))
mse_knn2 = mean_square_errors(((y_true=  y_test_df1  ; y_pred=y_pred_knn2)
print(mse_kn2/1e6)
```

Output: 159388.539

Test Output - Not Found	Train Output - Not Found	26349.994	218161.663

y_train

```
Perform Prediction for training data
```

y_pred_rf = rf2.predct(| X_train_df2)

accuracy_rf = accuracy_scores(y_true=y_trains_df2, y_pred= y_pred_rf))

print(accuracy_rf))

Output: 0.670

```
Performing Prediction for test data
```

y_pred_rf2 = rf2.predict(X_test_df2)

accuracy_rf2 = accuracy_scores(y_true= y_test_df2 , y_preds=y_pred_rf2

print(accuracy_rf2)

output: 0.615

0.718 | y_pred_rf2 | Test Output - Not Found

0.660 | Train Output - Not Found