MBA Program
Data Science
Assignment #2
Majdal Hindi
300991890

1. Use the Vehicle dataset CAR DETAILS FROM CAR DEKHO.csv to build a linear regression model that predicts the "selling_price" for each car. Split the data to 70% train and 30% test and report your results.

a. Which variables did you use? Try to select a subset of the variables and check whether you get better results.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4340 entries, 1 to 4340
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   brand             4340 non-null   object
 1   year              4340 non-null   int64 
 2   selling_price     4340 non-null   int64 
 3   km_driven         4340 non-null   int64 
 4   fuel              4340 non-null   object
 5   seller_type       4340 non-null   object
 6   transmission      4340 non-null   object
 7   owner             4340 non-null   object
 8   #_owner_children  4340 non-null   int64 
dtypes: int64(4), object(5)
memory usage: 339.1+ KB
```

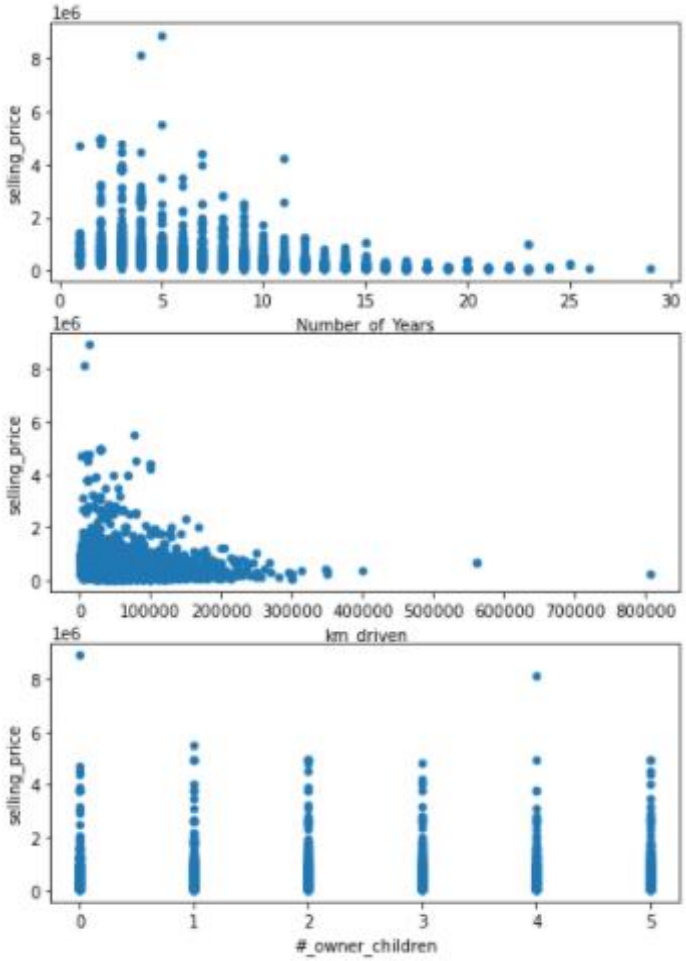
From the dataset, we can tell that:

1. There is no null values.
2. we can find that there are four numerical variables:

   *The target variable is selling_price
   *The numerical features are year, km_driven, #_owner children.

The 'year' column is numerical but is not representative of the type of relationship with the target column. By transforming it, we can create a new feature by creating [number of years] = current year (2021) -year (manufacturing year)

The graph below shows the relationship between the numerical features and the target variable:



From the graph, we can tell that the **year** feature correlates the most with the **selling price** column. The relationship is negative between the selling price and the number of years, km_driven variables. There is almost no relationship between the #_owner children and the **selling_price**. Therefore, I dropped the #_owner children column from the data set.

3. We can find that there are five categorical variables; to use these features in our model, we need to transform them into numerical representations.

The correlation between the features and the target in the final data frame:

```
brand_Skoda                        0.003078
brand_Force                        0.004150
fuel_Electric                      0.005095
brand_Volkswagen                   0.009094
brand_Nissan                       0.011460
brand_Daewoo                       0.011655
brand_Honda                        0.014601
brand_OpelCorsa                    0.015427
brand_Kia                          0.020886
brand_Mitsubishi                   0.021603
brand_Isuzu                        0.026135
brand_Renault                      0.027335
brand_Ford                         0.027959
brand_Datsun                       0.033198
brand_Fiat                         0.035377
fuel_LPG                           0.042434
brand_Mahindra                     0.043783
brand_Jeep                         0.046641
owner_Test_Drive_Car               0.048799
brand_MG                           0.049677
brand_Hyundai                      0.070294
Fourth_Above_Owner                 0.078725
brand_Chevrolet                    0.098361
brand_Jaguar                       0.100662
brand_Volvo                        0.107745
seller_type_Trustmark_Dealer       0.110176
Third_Owner                        0.111326
brand_Tata                         0.114574
Second_Owner                       0.161986
brand_Toyota                       0.162161
brand_Maruti                       0.180798
brand_Land                         0.182329
km_driven                          0.192289
seller_type_Individual             0.236798
fuel_Petrol                        0.269453
fuel_Diesel                        0.282947
brand_Audi                         0.292174
brand_Mercedes_Benz                0.354215
brand_BMW                          0.401857
Number_of_Years                    0.413922
transmission_Manual                0.530205
selling_price                      1.000000
Name: selling_price, dtype: float64
```

After splitting the data to train and test sets, My Simple linear regression model:

selling_*price*=a0+a1×Number_of_Years

a0=986,000, a1= -59,780, R^2= 0.177

The fitted model can be represented as:

selling_*price*= 986,000- 59,780 × **Number_of_Years**

The interruption of the model:

For every 1-year increase in the age of the car, we expect the car's selling price to decrease by approximately 59,780 dollars"

Model #1

OLS Regression Results

Dep. Variable:	selling_price	R-squared:	0.177
Model:	OLS	Adj. R-squared:	0.176
Method:	Least Squares	F-statistic:	650.7
Date:	Sat, 15 May 2021	Prob (F-statistic):	3.22e-130
Time:	14:37:51	Log-Likelihood:	-44423.
No. Observations:	3038	AIC:	8.885e+04
Df Residuals:	3036	BIC:	8.886e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9.86e+05	2.1e+04	47.019	0.000	9.45e+05	1.03e+06
Number_of_Years	-5.978e+04	2343.320	-25.510	0.000	-6.44e+04	-5.52e+04

Omnibus:	2967.901	Durbin-Watson:	1.778
Prob(Omnibus):	0.000	Jarque-Bera (JB):	145510.649
Skew:	4.749	Prob(JB):	0.00
Kurtosis:	35.547	Cond. No.	19.3

The Multiple linear regression model (using the numerical features and the dummy features):

Note: I dropped the features, which have low correlation with the selling price to build a linear regression model with better evaluation measures, but the best model I got when I use all the features as shown below.

Model #2

OLS Regression Results

Dep. Variable:	selling_price	R-squared:	0.734
Model:	OLS	Adj. R-squared:	0.731
Method:	Least Squares	F-statistic:	212.5
Date:	Sun, 16 May 2021	Prob (F-statistic):	0.00
Time:	09:42:38	Log-Likelihood:	-42705.
No. Observations:	3038	AIC:	8.549e+04
Df Residuals:	2998	BIC:	8.573e+04
Df Model:	39		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.036e+06	3.21e+05	3.226	0.001	4.06e+05	1.67e+06
km_driven	-0.9869	0.151	-6.529	0.000	-1.283	-0.691
Number_of_Years	-3.933e+04	1713.436	-22.955	0.000	-4.27e+04	-3.6e+04
brand_Audi	9.581e+05	3.17e+05	3.022	0.003	3.36e+05	1.58e+06
brand_BMW	2.093e+06	3.19e+05	6.570	0.000	1.47e+06	2.72e+06
brand_Chevrolet	-1.548e+05	3.14e+05	-0.493	0.622	-7.71e+05	4.61e+05
brand_Daewoo	1.321e+05	4.41e+05	0.300	0.764	-7.32e+05	9.96e+05
brand_Datsun	-2.276e+05	3.2e+05	-0.711	0.477	-8.55e+05	4e+05
brand_Fiat	-1.365e+05	3.2e+05	-0.427	0.669	-7.63e+05	4.9e+05
brand_Force	-2.507e+05	4.41e+05	-0.569	0.569	-1.11e+06	6.13e+05
brand_Ford	3.612e+04	3.14e+05	0.115	0.908	-5.8e+05	6.52e+05
brand_Honda	3.901e+04	3.14e+05	0.124	0.901	-5.77e+05	6.55e+05
brand_Hyundai	-2.993e+04	3.13e+05	-0.096	0.924	-6.44e+05	5.84e+05
brand_Isuzu	7.484e+05	4.41e+05	1.698	0.090	-1.16e+05	1.61e+06
brand_Jaguar	1.376e+06	3.38e+05	4.065	0.000	7.12e+05	2.04e+06
brand_Jeep	8.611e+05	3.83e+05	2.251	0.024	1.11e+05	1.61e+06
brand_Kia	-1.615e-09	8.59e-10	-1.880	0.060	-3.3e-09	6.97e-11
brand_Land	2.881e+06	3.61e+05	7.973	0.000	2.17e+06	3.59e+06
brand_MG	9.993e+05	4.41e+05	2.266	0.024	1.35e+05	1.86e+06
brand_Mahindra	6.327e+04	3.13e+05	0.202	0.840	-5.51e+05	6.78e+05
brand_Maruti	-6.526e+04	3.13e+05	-0.208	0.835	-6.79e+05	5.49e+05
brand_Mercedes_Benz	2.094e+06	3.19e+05	6.570	0.000	1.47e+06	2.72e+06
brand_Mitsubishi	4.16e+05	3.42e+05	1.215	0.225	-2.56e+05	1.09e+06
brand_Nissan	-1.682e+04	3.17e+05	-0.053	0.958	-6.39e+05	6.05e+05
brand_OpelCorsa	1.514e+05	4.39e+05	0.345	0.730	-7.09e+05	1.01e+06
brand_Renault	-1.477e+05	3.15e+05	-0.470	0.639	-7.65e+05	4.69e+05
brand_Skoda	-4.359e+04	3.16e+05	-0.138	0.890	-6.63e+05	5.76e+05
brand_Tata	-1.403e+05	3.13e+05	-0.448	0.654	-7.55e+05	4.74e+05
brand_Toyota	4.148e+05	3.14e+05	1.321	0.187	-2.01e+05	1.03e+06
brand_Volkswagen	-8.104e+04	3.15e+05	-0.257	0.797	-6.99e+05	5.37e+05
brand_Volvo	1.037e+06	3.83e+05	2.709	0.007	2.87e+05	1.79e+06
fuel_Diesel	1.901e+05	6.45e+04	2.948	0.003	6.37e+04	3.17e+05
fuel_Electric	-1.799e-10	1.82e-10	-0.986	0.324	-5.38e-10	1.78e-10
fuel_LPG	4.174e+04	9.68e+04	0.431	0.666	-1.48e+05	2.32e+05
fuel_Petrol	5664.2942	6.41e+04	0.088	0.930	-1.2e+05	1.31e+05
seller_type_Individual	-9119.4539	1.48e+04	-0.615	0.539	-3.82e+04	2e+04
seller_type_Trustmark_Dealer	2.75e+05	3.49e+04	7.891	0.000	2.07e+05	3.43e+05
transmission_Manual	-3.078e+05	2.3e+04	-13.371	0.000	-3.53e+05	-2.63e+05
Fourth_Above_Owner	6697.8800	4.28e+04	0.156	0.876	-7.73e+04	9.07e+04
Second_Owner	-4.388e+04	1.46e+04	-2.999	0.003	-7.26e+04	-1.52e+04
owner_Test_Drive_Car	1.669e+05	8.16e+04	2.044	0.041	6792.110	3.27e+05
Third_Owner	-2.243e+04	2.45e+04	-0.915	0.360	-7.05e+04	2.57e+04

Omnibus:	1965.575	Durbin-Watson:	2.030
Prob(Omnibus):	0.000	Jarque-Bera (JB):	156240.527
Skew:	2.276	Prob(JB):	0.00
Kurtosis:	37.836	Cond. No.	1.23e+16

b. Which evaluation measures did you use? Why? Describe your results and their meaning.

Regression Metrics for Model #1:

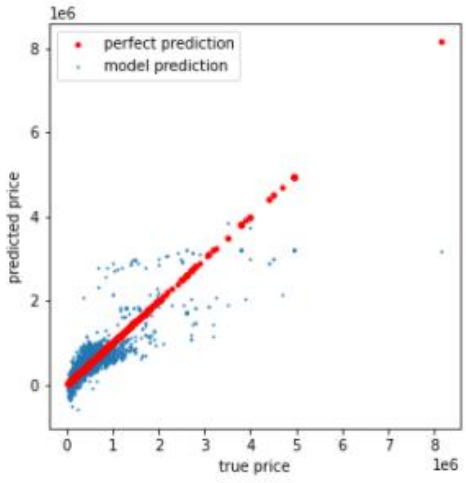
```
Regression finished with R^2=0.176507
Pearson coefficient: correlation between true and predicted: 0.4201
Regression accuracy (R^2) estimated via pearson coefficient: 0.1765
train_MSE: 294085676037.0417
test_MSE: 238950612683.5562
train_RMSE: 542296.6679199142
test_RMSE: 488825.7487935309
Mean absolute percentage error: 83.488
Mean absolute error: 254709.189
```

Regression Metrics for Model #2:

```
Regression finished with R^2=0.734311
Pearson coefficient: correlation between true and predicted: 0.8569
Regression accuracy (R^2) estimated via pearson coefficient: 0.7343
train_MSE: 94882904869.3391
test_MSE: 129892824130.58806
train_RMSE: 308030.6881941134
test_RMSE: 360406.4707113179
Mean absolute percentage error: 58.883
Mean absolute error: 182195.493
```

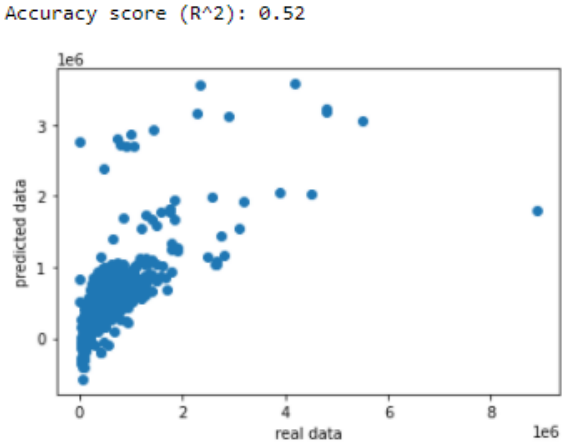
From the analysis above, we can see that all the values of the evaluation measures of model 2 are better than model 1, lets explain each metrics:

1- **R-squared** based on correlation between actual and predicted value. Higher R-squared value represent smaller differences between the observed data and the fitted values. In our model, R squared (on the train set) value tells us that about 73.43% of the total variation about the Selling price´ mean is explained by the regression line.
When a regression model accounts for more of the variance, the data points are closer to the regression line. In case, the fitted values equal the data values and, consequently, all the observations fall exactly on the regression line.



2-Pearson coefficient: It shows the linear relationship between two sets of data (true and predicted data). **Pearson Coefficient (on the train set)** =0.8569, was statistically highly significantly different from zero.

The graph below shows the Pearson coefficient on the test set = 0.52



3-MAE=182,195.493 implies that, on average, the forecast's distance from the true value is 182,195.493 e.g. True value is 150000 and forecast is 150,000+or -182195.493.

4- **MAPE**=58.883 implies that, on average, the forecast's distance from the true value is 58.883 % of the true value (e.g true value is 100,000 and forecast is 158883 or true value is 100,000 and forecast is 41,117 would be 58.883%).

5-MSE reflects the bias; the bias describes an error that results in bad assumptions about the learning algorithm. The features that we used in the model led to fit univariate regression model that results in high bias. High MSE means The error rate is high.

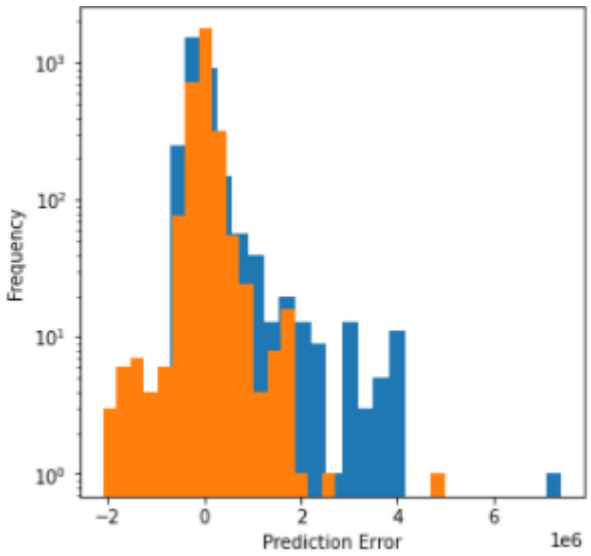
-MSE of the train set= 94,882,904,869 is used to check how close estimates or forecasts are to actual values. Lower the MSE, the closer is forecast to actual, the lower value indicates a better fit.

-MSE of the test set= 129,892,824,130

RMSE, is the square root of the MSE of all the error. It is considered an excellent general-purpose error metric for numerical predictions. It tells how concentrated the data is around the line of best fit

-RMSE of the train set=308,030
-RMSE of the test set=360,406

The graph below shows the prediction error in model#1 (in blue), and the prediction error in Model #2 (in orange). We can see that model #2 acts better than Model #1.



The table below shows the predicted price of the train set:

id	selling_price	km_driven	Number_of_Years	predicted Price
1	60000	70000	14	39575.905225
2	135000	50000	14	59313.665617
3	600000	100000	9	426435.014399

The table below shows the predicted price of the test set:

id	selling_price	km_driven	Number_of_Years	predicted Price
4338	110000	83000	12	61533.928807
4339	865000	90000	5	593634.171331
4340	225000	40000	5	340687.657884

c. Predict the selling price for the new cars in stock presented in Table 1.

By using model #2:

	km_driven	predicted_price	brand	year	fuel	seller_type	transmission	owner
0	40000	839833.0	Honda	2017	Petrol	Dealer	Automatic	Second Owner
1	6000	508678.0	Hyundai Creta	2016	Petrol	Dealer	Manual	Third Owner
2	9500	2766574.0	Mercedes-Benz	2015	Diesel	Dealer	Manual	First Owner

2. Build both logistic regression model and a decision tree model to predict whether the car brand be sold above their brand's mode price or not.

a. Describe your experiment, the results, and their meaning.

1- The table below shows the mode price of each brand

brand	mode
Ambassador	120000
Audi	1580000
BMW	4950000
Chevrolet	130000
Daewoo	60000
Datsun	250000
Fiat	350000
Force	346000
Ford	650000
Honda	300000
Hyundai	550000
Isuzu	1500000
Jaguar	2050000
Jeep	1400000
Kia	1300000
Land	4000000
MG	1825000
Mahindra	300000
Maruti	250000
Mercedes-Benz	3800000
Mitsubishi	825000
Nissan	300000
OpelCorsa	142000
Renault	300000
Skoda	385000
Tata	150000
Toyota	2600000
Volkswagen	350000
Volvo	1975000

2- I added the mode price column and the Boolean column ¨price above mode as shown in the table below:

id	brand	year	selling_price	km_driven	fuel	seller_type	transmission	owner	#_owner_children	mode_price	price_above_mode
1	Maruti	2007	60000	70000	Petrol	Individual	Manual	First Owner	3	250000.0	False
2	Maruti	2007	135000	50000	Petrol	Individual	Manual	First Owner	0	250000.0	False
3	Hyundai	2012	600000	100000	Diesel	Individual	Manual	First Owner	4	550000.0	True
4	Datsun	2017	250000	46000	Petrol	Individual	Manual	First Owner	5	250000.0	False
5	Honda	2014	450000	141000	Diesel	Individual	Manual	Second Owner	2	300000.0	True

3- Again, to use categorical features in logistic regression model, we need to transform them into numerical representations (including the year feature).

The features cols:

```
cols= ['brand_Ambassador', 'brand_Audi', 'brand_BMW', 'brand_Chevrolet',
       'brand_Daewoo', 'brand_Datsun', 'brand_Fiat', 'brand_Force',
       'brand_Ford', 'brand_Honda', 'brand_Hyundai', 'brand_Isuzu',
       'brand_Jaguar', 'brand_Jeep', 'brand_Kia', 'brand_Land', 'brand_MG',
       'brand_Mahindra', 'brand_Maruti', 'brand_Mercedes-Benz',
       'brand_Mitsubishi', 'brand_Nissan', 'brand_OpelCorsa', 'brand_Renault',
       'brand_Skoda', 'brand_Tata', 'brand_Toyota', 'brand_Volkswagen',
       'brand_Volvo', 'year_1992', 'year_1995', 'year_1996', 'year_1997',
       'year_1998', 'year_1999', 'year_2000', 'year_2001', 'year_2002',
       'year_2003', 'year_2004', 'year_2005', 'year_2006', 'year_2007',
       'year_2008', 'year_2009', 'year_2010', 'year_2011', 'year_2012',
       'year_2013', 'year_2014', 'year_2015', 'year_2016', 'year_2017',
       'year_2018', 'year_2019', 'year_2020', 'fuel_CNG', 'fuel_Diesel',
       'fuel_Electric', 'fuel_LPG', 'fuel_Petrol', 'seller_type_Dealer',
       'seller_type_Individual', 'seller_type_Trustmark_Dealer',
       'transmission_Automatic', 'transmission_Manual', 'owner_First',
       'Fourth_Above_Owner', 'owner_Second ',
       'owner_Test_Drive_Car', 'owner_Third']
```

4- Our new target variable is ¨price above mode¨.

5- Building the logistic regression: R^2= 0.843318, the logistic regression model works better than the linear regression model using the categorical features. (on all the dataset).Note: dropping the km_driven feature gave us better values

```
logistic_model = LogisticRegression()
y = df['price_above_mode']
X = df[cols]
logistic_model.fit(X,y)
print('Regression finished with R^2={0:f}'.format(logistic_model.score(X,y)))
```

```
Regression finished with R^2=0.843318
```

6 -Dividing the data set to train and test sets.

```
Training set size: 3047
Test set size: 1293
```

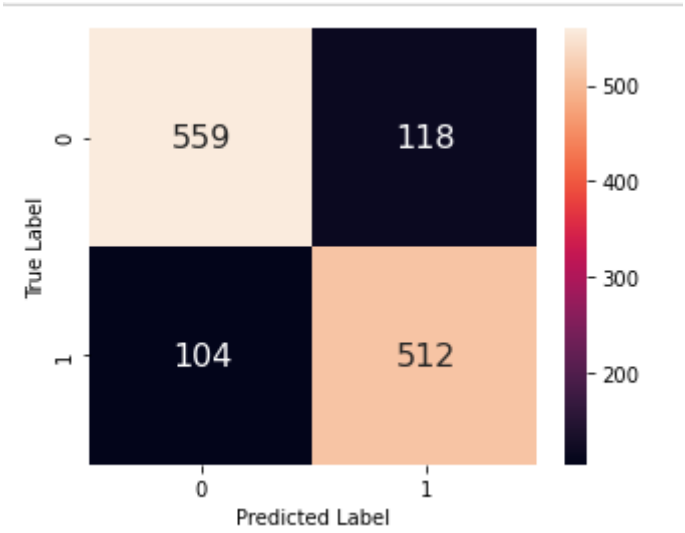
7-Fitting the training set by using the logistic model.

```
lr = LogisticRegression(solver='lbfgs', max_iter=100)
model=lr.fit(X_train,y_train)
```

a- calculating the evaluation measures of the logistic model

```
Precision score: 0.813
Recall score: 0.831
F1 score: 0.822
AUC score: 0.828
```

Lets interpt the meaning of each metrics from the confusion matrix in the test set:



The price above mode takes a ‘True ‘value
The price below mode takes a ‘False’ value

TP=512, we predict that the selling price will be above brand mode and the car sold above mode.
FP=118, we predict that the selling price will be above brand mode and the car sold below mode.
TN=559, we predict that the selling price will be below brand mode and the car sold below mode.
FN=104, we predict that the selling price will be below brand mode and the car sold above mode.

Calculating the metrics:

Precision= 512/(512+118)=0.813
Recall=512/(512+104)=0.831
Sensitivity=512/(512+104)=0.831
Specificity =559/(559+118)=0.838
Accuracy=(512+559)/(512+559+118+104)=0.828

Interruption of the metrics:

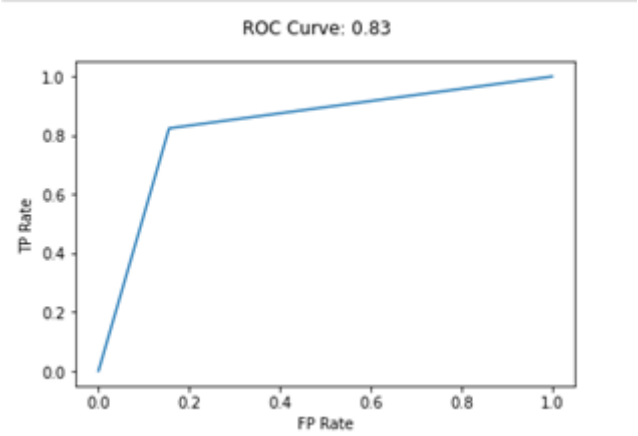
The accuracy means is the proportion of correct predictions (TP+TN) over total predictions (test set size), in our case the AUC=82.8% which is significantly high. Accuracy is a great measure but only when we have values of false positive and false negatives are almost same. For our model, we have 0.828 which means our model is approx. 82.8% accurate.

Precision = 81.3% - fraction of correctly predicted ‘price above mode’ among the ones marked as ‘price above mode’ by the classifier. High precision relates to the low false positive rate (wrong predicted True values). The question that this metric answer is of all car sold that the selling price labeled as above mode, how many cars actually sold above mode?

Recall = 83.1% fraction of identified ‘price above mode’ between all ‘price above mode and price below mode’. Also called Sensitivity. The question recall answers is: Of all car sold which ones had a selling price above mode, how many did we label?

The graph below shows the precision-recall curve; the shape of the curve contains the expected false positive rate, and the false negative rate.

Smaller values on the x-axis of the plot indicate lower false positives and higher true negatives.Larger values on the y-axis of the plot indicate higher true positives and lower false negatives.



In the classification, report below we can notice a new metric F1-score.F1 Score is the weighted average of Precision and Recall. In our case, F1 score is 0.83, which is significantly high.

	precision	recall	f1-score	support
False	0.84	0.83	0.83	677
True	0.81	0.83	0.82	616
accuracy			0.83	1293
macro avg	0.83	0.83	0.83	1293
weighted avg	0.83	0.83	0.83	1293

8- Fitting the training set by using the decision Tree model, and calculating the evaluation measures:

```
clf = DecisionTreeClassifier(random_state=1)
clf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(random_state=1)
```

```
from sklearn.metrics import roc_auc_score
predictions = clf.predict(X_test)
error=roc_auc_score(y_test,predictions)
print('test:',error)
predictions = clf.predict(X_train)
print('train:',roc_auc_score(y_train,predictions))
```

```
test: 0.8390552995391706
train: 0.9054494856732739
```

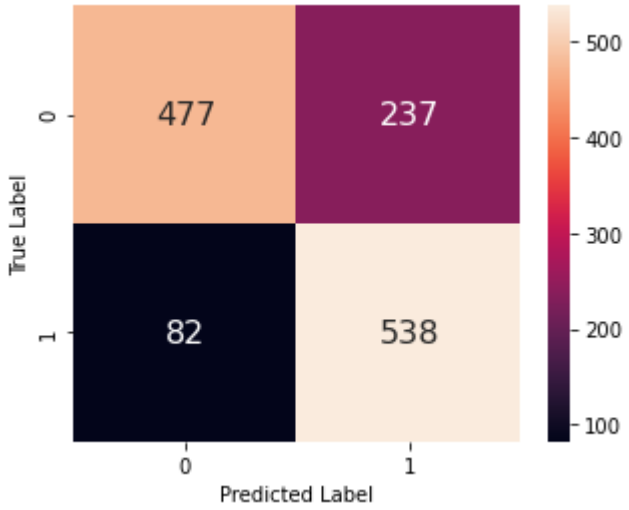
We can notice that there is overfitting, so I used 13 sample split and the depth =7 in the Decision Tree model. I got better results of AUC

```
clf1 = DecisionTreeClassifier(random_state=1, min_samples_split=13, max_depth=7)
clf1.fit(X_train, y_train)
predictions = clf1.predict(X_test)
test_auc = roc_auc_score(y_test, predictions)
train_predictions = clf1.predict(X_train)
train_auc = roc_auc_score(y_train, train_predictions)
```

```
test_auc: 0.7679045811873136
train_auc 0.7797359777687818
```

The evaluation metrics:

```
Precision score: 0.694
Recall score: 0.868
F1 score: 0.771
AUC score: 0.768
```



b-which model perfumed better on your test set?

Logistic Model measures:

```
Precision score: 0.813
Recall score: 0.831
F1 score: 0.822
AUC score: 0.828
```

Decision Tree Model:

```
Precision score: 0.694
Recall score: 0.868
F1 score: 0.771
AUC score: 0.768
```

We can notice that the values of the logistic Model are better than The Decision Tree Model.

c.Use your model to predict whether the new cars in stock presented in Table 1 be sold above their brand's mode price.

Prediction-Logistic Regression Model

	km_driven	mode_price	price_above_mode_logistic	brand	year	fuel	seller_type	transmission	owner
0	40000	300000.0	True	Honda	2017	Petrol	Dealer	Automatic	Second Owner
1	6000	550000.0	False	Hyundai	2016	Petrol	Dealer	Manual	Third Owner
2	9500	3800000.0	False	Mercedes-Benz	2015	Diesel	Dealer	Manual	First Owner

Hyundai and Mercedes-Benz sold below the brand mode price; Honda sold above the brand mode price, by using the logistic model.

Prediction-Decision Tree Model

	km_driven	mode_price	price_above_mode_tree	brand	year	fuel	seller_type	transmission	owner
0	40000	300000.0	False	Honda	2017	Petrol	Dealer	Automatic	Second Owner
1	6000	550000.0	False	Hyundai	2016	Petrol	Dealer	Manual	Third Owner
2	9500	3800000.0	True	Mercedes-Benz	2015	Diesel	Dealer	Manual	First Owner

Hyundai and Honda sold below the brand mode price; Mercedes-Benz sold above the brand mode price, by using the Decision Tree model.

d. Repeat your experiment with 5-folds cross validation and report your results. Are they different from your 70%-30% split?

```
from sklearn.model_selection import cross_val_score
y_data=df['price_above_mode']
X_data=df.drop(columns=['selling_price', 'km_driven', 'mode_price', 'price_above_mode'])
model=LogisticRegression(solver='lbfgs', max_iter=10000)
clf1 = DecisionTreeClassifier(random_state=1, min_samples_split=13, max_depth=7)
scores_logistic = cross_val_score(model, X_data, y_data, cv=5)
scores_tree = cross_val_score(clf1, X_data, y_data, cv=5)
print("Accuracy_logistic: %0.2f (+/- %0.2f)" % (scores_logistic.mean(), scores.std() * 2))
print("Accuracy_tree: %0.2f (+/- %0.2f)" % (scores_tree.mean(), scores.std() * 2))
```

```
Accuracy_logistic: 0.83 (+/- 0.01)
Accuracy_tree: 0.76 (+/- 0.01)
```

There is no difference between the accuracy of the two models by using 5-folds cross validation and the 70-30% split.

3. **(40%) Build a decision tree model to predict whether the car be sold at their brand's 25th percentile,50th percentile, 75th percentile or top percentile.**

a-Where all variables used by the tree?

In the depth =32 which is the length of the longest path from a root to a leaf.

b. Show your tree.

```
display_tree(clf)
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.54598 to fit
```

```
print(f"The tree has {clf.get_n_leaves()} leafs, {clf.tree_.node_count} nodes and deapth of {clf.get_depth()}")
```

```
The tree has 1169 leafs, 2337 nodes and deapth of 32
```

The display of the tree after trying to decrease the overfit by setting min_samples_split to 13

```
The tree has 473 leafs, 945 nodes and deapth of 28
```

```
display_tree(clf)
```

The text representation of the tree:

Press the link: decistion_tree.log

c. Describe your results or your model.

After transforming and selecting the features that we want to use in our model. We want to predict if the car sold at their brand's 25th percentile, 50th percentile, 75th percentile or top percentile. To do that, we should calculate the percentile of the selling price for each brand, then to divide the result to 4 lots. In our case Q1, Q2,Q3,Q4.

Q1: if the selling_price < 0.25_selling_price_percentile for each brand
Q2: if the selling_price < 0.5_selling_price_percentile for each brand
Q3: if the selling_price < 0.75_selling_price_percentile for each brand
Q4: otherwise

After setting these conditions, we get a new target variable selling_price_percentile

The tabe below is the Calculations of the percentile for each brand:

```
{'Ambassador': 0.25    102500.0
0.50    120000.0
0.75    197500.0
Name: selling_price, dtype: float64, 'Audi': 0.25    1300000.0
0.50    1580000.0
0.75    1850000.0
Name: selling_price, dtype: float64, 'BMW': 0.25    1520000.0
0.50    2600000.0
0.75    4950000.0
Name: selling_price, dtype: float64, 'Chevrolet': 0.25    150000.0
0.50    200000.0
0.75    290000.0
Name: selling_price, dtype: float64, 'Daewoo': 0.25    60000.0
0.50    60000.0
0.75    60000.0
```

Creating a new Target Variable:

id	year	selling_price	km_driven	brand_Ambassador	brand_Audi	brand_BMW	brand_Chevrolet	brand_Daewoo	brand_Datsun	brand_Fiat	...
1	2007	Q1	70000	0	0	0	0	0	0	0	...
2	2007	Q1	50000	0	0	0	0	0	0	0	...
3	2012	Q4	100000	0	0	0	0	0	0	0	...
4	2017	Q2	46000	0	0	0	0	0	1	0	...
5	2014	Q2	141000	0	0	0	0	0	0	0	...

5 rows × 48 columns

After splitting the dataset to train and test sets, fitting the train sets using the decision tree model, and we got these results of the evaluation measures:

```
Training Set Classification Accuracy: 0.96
              precision    recall  f1-score   support

          Q1       0.99      0.96      0.97       769
          Q2       0.96      0.94      0.95       777
          Q3       0.92      0.95      0.94       694
          Q4       0.96      0.98      0.97       800

    accuracy                           0.96      3040
   macro avg       0.96      0.96      0.96      3040
weighted avg       0.96      0.96      0.96      3040


Test Set Classification Accuracy: 0.60
              precision    recall  f1-score   support

          Q1       0.73      0.67      0.70       334
          Q2       0.51      0.48      0.50       330
          Q3       0.48      0.52      0.50       296
          Q4       0.68      0.71      0.69       343

    accuracy                           0.60      1303
   macro avg       0.60      0.60      0.60      1303
weighted avg       0.60      0.60      0.60      1303
```

The accuracy, F1-score on the test set = 60%, which is good. We can see also from the table above, the accuracy, F1-score of predicting Q1 lot is the highest, and the accuracy, F1-score of predicting Q3 is the lowest.

The classifier predictions performance on the train set:

brand	year	selling_price	km_driven	fuel	seller_type	transmission	owner	predictions
Maruti	2007	Q1	70000	Petrol	Individual	Manual	First Owner	Q1
Maruti	2007	Q1	50000	Petrol	Individual	Manual	First Owner	Q1
Hyundai	2012	Q4	100000	Diesel	Individual	Manual	First Owner	Q4
Datsun	2017	Q2	46000	Petrol	Individual	Manual	First Owner	Q2
Honda	2014	Q2	141000	Diesel	Individual	Manual	Second Owner	Q2

The classifier predictions performance on the test set:

brand	year	selling_price	km_driven	fuel	seller_type	transmission	owner	predictions
Maruti	2012	Q2	90000	Diesel	Individual	Manual	Second Owner	Q3
Toyota	2012	Q2	170000	Diesel	Individual	Manual	First Owner	Q3
Hyundai	2014	Q3	80000	Diesel	Individual	Manual	Second Owner	Q3
Hyundai	2014	Q3	80000	Diesel	Individual	Manual	Second Owner	Q3
Maruti	2009	Q1	83000	Petrol	Individual	Manual	Second Owner	Q1
Hyundai	2016	Q4	90000	Diesel	Individual	Manual	First Owner	Q4

We can notice that there is overfitting, the model works better on the train sets than the test set

I tried to decrease the overfit by setting min_samples_split to 13, I got a version showing that the model we built was less overfit to the training set than before However, the version that I got actually has lower accuracy on our training set.

The table below shows the matrix confusion on the test set:

	Predicted Q1	Predicted Q2	Predict Q3	Predict Q4
True Q1	229	51	18	7
True Q2	70	161	61	21
True Q3	35	77	141	71
True Q4	16	45	66	234

The table below shows sample #2 of a sold car:

```
Prediction for sample #2 (year                          2007
km_driven                      50000
brand_Ambassador                   0
brand_Audi                         0
brand_BMW                          0
brand_Chevrolet                    0
brand_Daewoo                       0
brand_Datsun                       0
brand_Fiat                         0
brand_Force                        0
brand_Ford                         0
brand_Honda                        0
brand_Hyundai                      0
brand_Hyundai Creta                0
brand_Isuzu                        0
brand_Jaguar                       0
brand_Jeep                         0
brand_Kia                          0
brand_Land                         0
brand_MG                           0
brand_Mahindra                     0
brand_Maruti                       1
brand_Mercedes-Benz                0
brand_Mitsubishi                   0
brand_Nissan                       0
brand_OpelCorsa                    0
brand_Renault                      0
brand_Skoda                        0
brand_Tata                         0
brand_Toyota                       0
```

```
brand_Volkswagen                    0
brand_Volvo                         0
fuel_CNG                            0
fuel_Diesel                         0
fuel_Electric                       0
fuel_LPG                            0
fuel_Petrol                         1
seller_type_Dealer                  0
seller_type_Individual              1
seller_type_Trustmark Dealer        0
transmission_Automatic              0
transmission_Manual                 1
owner_First Owner                   1
owner_Fourth & Above Owner          0
owner_Second Owner                  0
owner_Test Drive Car                0
owner_Third Owner                   0
Name: 2, dtype: int64)
        Predicted: ['Q1']. Expected:Q1
        Predicted probabilities:[[0.9 0.1 0.  0. ]]

Overall prediction score:0.587107
```

c. Use your model to predict at which percentile the new cars in stock presented in Table 1 be sold.

	id	brand	year	km_driven	fuel	seller_type	transmission	owner	predictions
0	1	Honda	2017	40000	Petrol	Dealer	Automatic	Second Owner	Q4
1	2	Hyundai Creta	2016	6000	Petrol	Dealer	Manual	Third Owner	Q3
2	3	Mercedes-Benz	2015	9500	Diesel	Dealer	Manual	First Owner	Q4

From the table, the model predictions:

Honda sold at top percentile selling price
Hyundai sold at 0.75 percentile selling price
Mercedes-Benz sold at a top percentile-selling price