Supervised Lab Assignment on predicting car price.

1) Load the file “imports-85.data” into a dataframe from https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.dataLinks to an external site. and column names of this data set can be found here --https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.namesLinks to an external site.

   Loading the file from above url into data of type dataframe. And getting the list of columns from above url into features list. And making list as column header of dataframe.

   Python Code:

   data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data',header=0)

   features = ['symboling','normalized_losses','make','fuel_type','aspiration','num_of_doors','body_style','drive_wheels','engine_location','wheel_base','length','width','height','curb_weight','engine_type','num_of_cylinders','engine_size','fuel_system','bore','stroke','compression_ratio','horsepower','peak_rpm','city_mpg','highway_mpg','price']

   data.columns = features

2) Explain the problem statement. What are you predicting and what attributes you have to predict?

   UCI database had used cars data which consist of

   a) The specification of automobiles
   b) Assigned insurance risk rating
   c) Normalized losses in use

   Data set has 204 records with 26 columns containing different features of cars among 15 are continues, 1 integer and 10 categorical variables.

   Missing attributes values:

   normalized-losses: 41 ,num-of-doors: 2,bore: 4,stroke: 4,horsepower:2,peak-rpm:2,price=4

   Target: Predicting price of a car using various predictors available in the dataset (price).

   Predictors: 25 attributes are available from the dataset to predict the price of a car.

   Ref: https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.names

3) Browse a sample record from the dataframe. Are there any missing values?

Browsing sample records from data or from UCI site we can see some missing values with '?' sign, Will replace with NaN which helps in processing(replacing with mean, median or mode).

Found 4 missing values in target (price) column, will remove those records.

Python Code:

```
data.head()
data.index[data['price'] == '?']
data.drop(data.index[[8,43,44,128]],axis=0,inplace=True)
data.replace('?',np.NaN,inplace=True)
data = data.apply(lambda x:x.fillna(x.value_counts().index[0]))
```

4) How many records are available in the data set and how many attributes. Do you think the depth (number of records) is sufficient given the breadth? In other words, is the sample likely to be a good representative of the universe?

204 records and 26 attributes are available in the dataset.
Among 26 attributed 16 are numeric and 10 are categorical.
Among 204, 4 records doesn't contain price value, So 200 records are available.

200 records with 25 attributes are sufficient when we use LableEncoder technique to convert categorical into numbers for predicting the price. But model will start considering the one categorical value greater than or less than the other.

But using get_dummies or one hot encoding technique we are suffer from curse of dimensionality as 10 categorical variables are converting into 50 dimensions.
For such cases our sample is not a good representative of universe as we fall under very less data points (200 in 65 dimensions). Further splitting data into training (140,65) and test sets(60,65) will worsen the condition so building model on such data set won't be helpful in predicating the price.

So LabelEncoding Technique is used in this case (200 * 26) to predict the target.

Python code:

```
from sklearn.preprocessing import LabelEncoder
data = data.apply(LabelEncoder().fit_transform)
```

5) Analyse the data distribution for the various attributes and share your observations.

data.describe() and seaborn library's distplot method helps in understanding and visualizing the distribution for various attributes.
Symbolling: 200 data points, ranges from -3 to +3, but min = -2 and max = +3 no data point with value -3.
mean 0.83 < median 1 indicates slightly skewed to left in 1st quartile.

normalized_losses: 200 data points ranges from 65 to 256. Mean and median are almost same. But the difference in 3rd and 4th quantile is more, Indicates long tail at the right. Data points 231 and 256 can be considered as outliers. Even distplot() visualization shows Gaussians, indicates values are generated by 2 different sources.

wheel_base: 200 data points ranges from 86 to 120.Mean and median are almost same. Dense at the center and skewed towards right as distance b/w 75% and max is more. Data point 120 can be treated as outlier.

Length: 200 data points ranges from 141 to 208. Mean and median are almost same. Have long tail on both the sides, normally distributed with 99.97% data covered within 3 standard deviation.

Width: 200 data points ranges from 60 to 72 with 2 standard deviation. Mean and median are almost same. Skewed towards right as distance between 3rd and 4th quartile is more. Distributed within 3 standard deviation.

Height: 200 data points ranges from 47 to 59. Skewed to the left so mean is slightly pulled to the left.

curb_weight: 200 data points ranges from 1488 to 4066. Mean is greater than median as body is skewed towards right. Data point 1488 can be treated as outlier.

engine_size: 200 data points ranges from 61 to 326. Mean is greater than median. Skewed towards rights with long tail towards right side Indicating outliers. Values 304,308 and 326 can be treated as outliers.

Bore: 200 data points ranges from 2 to 4.Mean is 1 SD less than median. Body skewed towards left with long tail. Few points can be considered as outliers to make it normal distribution.

Stroke: 200 data points ranges from 2.07 to 4.17. mean is slightly less than median so body is skewed towards left. Long tail on both sides.

Horsepower: 200 data points ranges from 48 to 262.Mean is greater than median. Very long tail at right side due to outlier data point 262. Data point need to be removed or replaced to achieve normal distribution.

city_mpg: 200 data points ranges from 13 to 49.Mean is slightly greater than median. Long tail towards right due to outliers. Mpg of car 45, 47 and 49 need to be taken care whether they are really claiming or due to some noise.

Python code:

```
data.describe()
sns.distplot( data.city_mpg )
```

6) Are there any independent attributes which have |R| close to 1?

Yes, pearson correlation coefficient helps in finding the linear relationship between 2 variables. corr() function and seaborn library methods helps in finding and visualizing the linear relationship.

city_mpg vs **highway_mpg**: **0.972** indicates strong relationship between this 2 independent variables.
While modeling either one or deriving new variable from 2 need to be considered with the help of PCA technique.
Other independent attributes which have linear relationship:

wheel_base vs length: 0.87

wheel_base vs width: 0.814

wheel_base vs curb_weight: 0.787

length vs width: 0.85

length vs curb_weight: 0.88

length vs highway_mpg: -0.7

width vs curb_weight: 0.86

width vs engine_size: 0.73

curb_weight vs engine_size: 0.84

curb_weight vs horsepower: 0.755

curb_weight vs city_mpg: -0.75

curb_weight vs highway_mpg: -0.79

engine_size vs horsepower: 0.81

python code:

```
data.engine_size.corr( data.city_mpg )
sns.jointplot( data.wheel_base , data.length )
```

data.corr(method='pearson', min_periods=1)

7) Which attributes seem to have a stronger relationship with the dependent variable (Price of the car)?

pearson correlation coefficient helps in finding the linear relationship between 2 variables. Now we will use independent vs dependent variables
corr() function and seaborn library methods helps in finding and visualizing the linear relationship.

Price vs curb_weight: 0.83

Price vs engine_size: 0.872

Price vs horsepower: 0.81

Price vs width: 0.75

Price vs length: 0.69

Price vs highway_mpg: -0.7

Price vs city_mpg: -0.68

Price vs fuel_system(post label encoding): 0.7

Taking the threshold of R (Pearson coefficient) value around ~7, with the help of EDA we can say above independent variable are good predictors of our target variable, price of a car.

Python code:

data.price.corr(data.fuel_system)
sns.jointplot(data.price , data.fuel_system)
data.corr(method='pearson', min_periods=1)

8) Given the above analysis, which algorithm will be more suitable? Why?

Please find the ipynb files attached with 4 different cases

Case 1: Applying get_dummies() on categorial variables and considering all the features.

Conclusion: model performance good on train set (overfitting) but fails on test set (curse of dimensionality)

Case 2: Applying LabelEncoder() on categorial variables and considering all the features

Conclusion: model performance good on train set compared to test set

Case 3: Applying LabelEncoder() on categorial variables and considering strong features whose R value close to 1

Conclusion: model performance good on both train as well as test set

Case 4: Using Stats model

Conclusion: Helps in finding Actual predictors which have $p<0.5$ and model parameters with R square and Adjusted R square.

With the help of Pearson R value we found the linear relationship between 2 variables (independent vs independent) and (independent vs dependent).

But the same variables in multidimensional space have different behavior, But using stats model we can identify the independent variables which are highly helpful in predicting the car price.

We tried using sklearn.LinearRegression() and stats model OLS() algorithms to find the model to predict the price of a car. As we didn’t go through the other algorithms in residency.