Configuration Manual

MSc Research Project
Data Analytics

Kishore Lakshmanan
Student ID: x20253583

School of Computing
National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kishore Lakshmanan
Student ID:	x20253583
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Mr. Hicham Rifai
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	835
Page Count:	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	13th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kishore Lakshmanan
x20253583

1 Introduction

The configuration manual gives the step-by-step guide to execute the modules which will be useful for this research project. The steps includes from software installation to model building process. This project comprises of two different stages such as fault identification part and another is price evaluation process. This manual contains code snippet as well to run the project without any problems.

2 System Configuration

2.1 Software Requirements

The research project was developed using the open source IDE called Jupyter Notebook which is available through the Anaconda software. This environment works based on python module. All these packages needs to be installed before building the project.

2.2 Hardware specifications

- System Name: DESKTOP-SM51BMP
- Processor: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, 2601 Mhz, 2 Core(s), 4 Logical Processor(s)
- Installed RAM: 8.00 GB
- Storage Size: 465.76 GB SSD (500,105,249,280 bytes)
- OS type: 64-bit operating system, x64-based processor

3 Installation and Environment Setup

- **Python**
  Python module was used in this project. Since, it has many in-build libraries which support most of the Deep Learning and Machine Learning Projects. It ease the model building and analyse with various plots. The first requirement is to install the latest version python in the system. Based on the operating system, the package installer can be downloaded from the website [1] through browser. After successful

[1]https://www.python.org/downloads/

installation of python from the website as shown below figure 1, type 'python -version' in the command prompt to verify it.

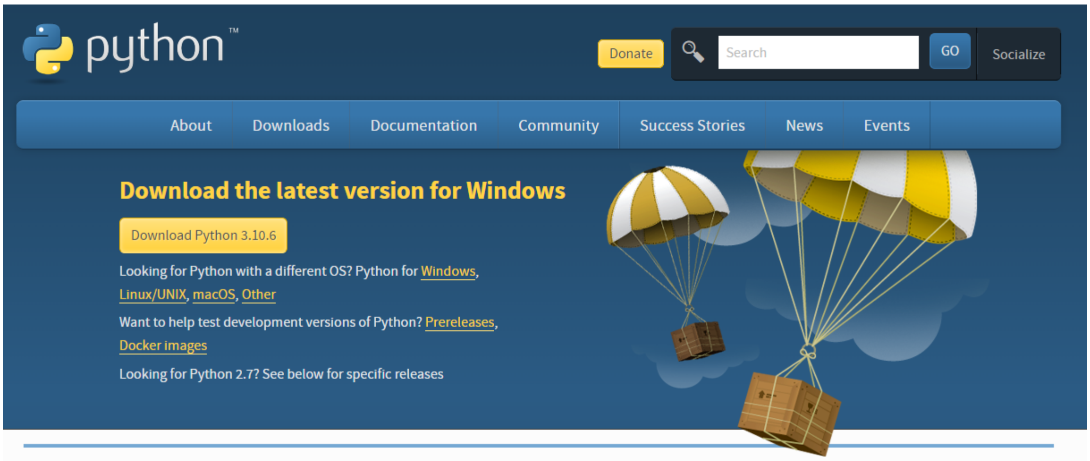


Figure 1: Python Website Page

- **Anaconda**
  Anaconda package comprises of several IDE which will be useful for developing the code and for analysing the outputs through the python package. This package can be downloaded and installed from the website [2]. There were lot of different IDE available in this navigator 2. In this project, Jupyter Notebook is used for building the model.

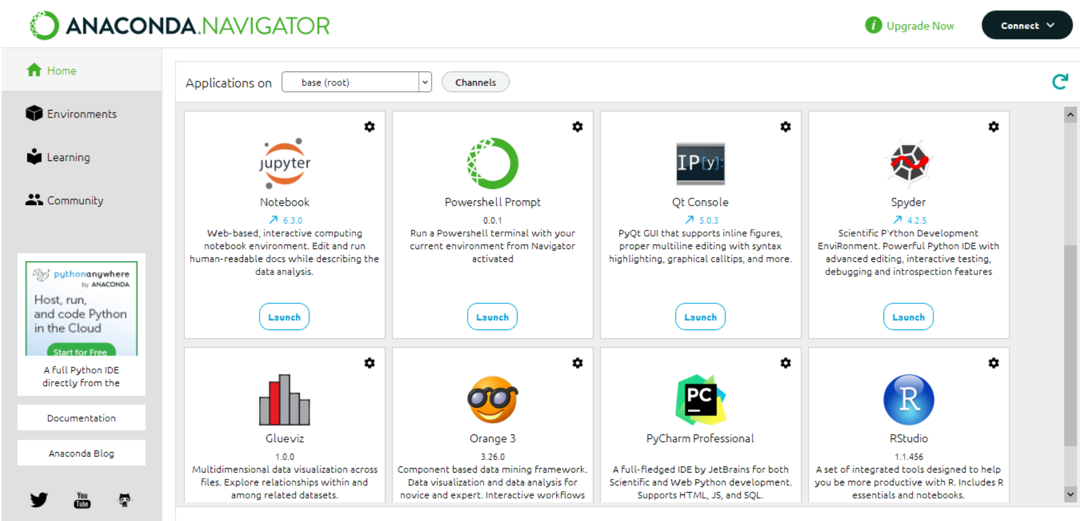


Figure 2: Anaconda Website Page

- **Jupyter Notebook**
  From the anaconda navigator, Jupyter notebook and it's tasks are launched in the browser tabs. Initially python notebook is created and saved as .ipynb format.

[2]https://www.anaconda.com/products/individual

The python libraries are installed during the implementation of code using pip command. The required libraries for this project are numpy, pandas, tensorflow, matplotlib, seaborn and plotly.

Command: pip install 'LibraryName'

4 Data Collection

There were two datasets used for this project which is taken from kaggle. Following sections are divided into two parts for building both defect detection [3] which contains the pre-processed image bunddle and cost evaluation [4] which includes the information about the vehicles. The output from each parts will be used to satisfy the project objectives.

5 Implementation of Defect Detection

5.1 Importing Libraries

Before implementation of model, the required libraries needs to be imported for smooth execution. The below figure 3 shows the imported libraries of our research.

```
In [1]: #Importing Libraries
        import numpy as np
        import pandas as pd
        import tensorflow as tf
        import os
        import matplotlib.pyplot as plt
        from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img, img_to_array
        from tensorflow.keras.preprocessing import image
        from keras.models import Sequential
        from tensorflow.keras.models import Model
        from tensorflow.keras.applications import MobileNet
        from keras.layers import Dense,Activation,Dropout, Conv2D, MaxPool2D
        from keras.callbacks import EarlyStopping, ModelCheckpoint

In [2]: print('NUMPY Version: %s' % np.__version__)
        print('PANDAS Version: %s' % pd.__version__)
        print('TENSORFLOW Version: {}'.format(tf.__version__))

        NUMPY Version: 1.20.1
        PANDAS Version: 1.2.4
        TENSORFLOW Version: 2.9.1
```

Figure 3: Importing Libraries for Defect Detection Model

5.2 Splitting of Train and Test Data

The dataset needs to be divided into training and validation data to develop the model as shown in the below figure 4

[3]https://www.kaggle.com/datasets/anujms/car-damage-detection
[4]https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho

```
In [3]: train_data = "D:/Masters in DA/Research Project/Datasets/Defect/training"
        test_data = "D:/Masters in DA/Research Project/Datasets/Defect/validation"

In [4]: img = plt.imread(os.path.join(train_data, "00-damage/0005.JPEG"))
        plt.imshow(img)
        height, width, dim = img.shape
        print("size of image (h x w)",height,width)

        size of image (h x w) 194 259
```

Figure 4: Splitting of Train and Test Data

5.3 Data Augmentation and Normalization

The data augmentation and normalization was done by the following code shown in the figure 18.

```
In [6]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

        train_gen = ImageDataGenerator(rescale=1/255)
        test_gen = ImageDataGenerator(rescale=1/255)

In [7]: train_dataset = train_gen.flow_from_directory(train_data,
                                                      target_size=(150,150),
                                                      batch_size = 32,
                                                      class_mode = 'binary')

        test_dataset = test_gen.flow_from_directory(test_data,
                                                    target_size=(150,150),
                                                    batch_size =32,
                                                    class_mode = 'binary')
        test_dataset.class_indices

        Found 1840 images belonging to 2 classes.
        Found 460 images belonging to 2 classes.

Out[7]: {'00-damage': 0, '01-whole': 1}
```

Figure 5: Data Augmentation and Normalization

5.4 CNN Model

The below code gives the overview of model building, setting hypertuning parameters and accuracy with graphs of CNN model.

Model Building

In [30]:
```
from keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.applications import MobileNet
from keras.layers import Dense,Activation,Dropout, Conv2D, MaxPool2D
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

In [9]:
```
# using CNN Model
model = Sequential()
```

In [10]:
```
# Convolutional layer & maxpool layer 1 to 4
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(MaxPool2D(2,2))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))

model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))

model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))
```

In [11]:
```
# Converting image array to 1D array
model.add(tf.keras.layers.Flatten())

# Activation function with relu and sigmoid
model.add(Dense(256,activation='relu'))
#model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))

model.summary()
```

Figure 6: CNN: Model Building Process

In [12]:
```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4, min_delta=0.001)
lr_reduce = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', verbose=1, patience=2, factor=0.1, min_lr=0.0001)
model_check = tf.keras.callbacks.ModelCheckpoint('best_model.hdf5', monitor='val_accuracy', verbose=1,save_best_only=True,mode='r
```

Training Model

In [30]:
```
model1 = model.fit(train_dataset,
                   validation_data=test_dataset,
                   epochs=20,
                   callbacks=[early_stop, lr_reduce, model_check],
                   batch_size=32)
```

```
Epoch 1/20
58/58 [==============================] - ETA: 0s - loss: 0.6933 - accuracy: 0.5364
Epoch 1: val_accuracy improved from -inf to 0.61522, saving model to best_model.hdf5
58/58 [==============================] - 63s 1s/step - loss: 0.6933 - accuracy: 0.5364 - val_loss: 0.6469 - val_accuracy: 0.615
2 - lr: 0.0010
```

Figure 7: CNN: Hyperparameter Tuning and Model Training

5.5 MobileNet Model

The below code gives the overview of model building, setting hypertuning parameters and accuracy with graphs of MobileNet model.

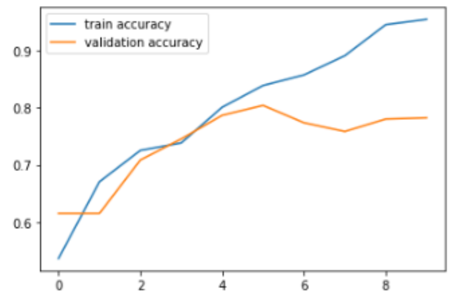
```
In [46]: def build_model():
             model = MobileNet(weights="imagenet",include_top=False, input_shape=(300, 300, 3), pooling='avg')
             for layer in model.layers[:-2]:
                 layer.trainable = False
                 predictions = Dense(2, activation="softmax")(model.output)
                 model = Model(inputs=model.input, outputs=predictions)
             return model

         model = build_model()

         WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape
         (224, 224) will be loaded as the default.

In [60]:

         model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

         early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='auto', verbose=1, patience=4, min_delta=0.001)
         lr_reduce = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', verbose=1, patience=2, factor=0.1, min_lr=0.0001)
         model_check = tf.keras.callbacks.ModelCheckpoint('MobileNet_Car_Classifier.h5', monitor='val_accuracy', verbose=1,save_best_only

In [61]: model1 = model.fit(train_dataset,
                            validation_data=test_dataset,
                            epochs=20,
                            callbacks=[early_stop, lr_reduce, model_check],
                            batch_size=32)

         Epoch 1/20
         58/58 [==============================] - ETA: 0s - loss: 0.6978 - accuracy: 0.5000
         Epoch 1: val_accuracy improved from -inf to 0.50000, saving model to MobileNet_Car_Classifier.h5
         58/58 [==============================] - 48s 736ms/step - loss: 0.6978 - accuracy: 0.5000 - val_loss: 0.6956 - val_accuracy: 0.
         5000 - lr: 0.0010
```

Figure 8: MobileNet: Model Building Process

5.6 Performance Analysis

The code shown below is given to find the accuracy of the CNN model and graph of train and test accuracy. This code can be used for both the model.

```
In [32]: plt.plot(model1.history['accuracy'], label='train accuracy')
         plt.plot(model1.history['val_accuracy'], label='validation accuracy')
         plt.legend()
         plt.show()
```




```
In [33]: print('Training accuracy achieved', model1.history['accuracy'][-2])

         Training accuracy achieved 0.945652186870575
```

Figure 9: CNN: Plot and Performance Analysis

5.7 Severity Assessment

The below code is given to find the prediction of car damage and severity level of the vehicle in the images.

```
In [90]:  from tensorflow.keras.preprocessing import image
          img=image.load_img('D:/Masters in DA/Research Project/Datasets/Defect/validation/00-damage/0001.jpeg', target_size=(150,150)
          #img = image.load_img("training",target_size=(150,150))
          plt.imshow(img)
          Y = np.array(img)
          X = np.expand_dims(Y,axis=0)
          val = model.predict(X)
          print(val)
          if val < 0.5:
              plt.xlabel("Car Damage",fontsize=25)
          elif val >= 0.5:
              plt.xlabel("Car Not Damage",fontsize=25)

1/1 [==============================] - 0s 42ms/step
[[1.4741634e-14]]
```

Figure 10: Prediction of Car Damage

```
In [91]:  img = load_img('D:/Masters in DA/Research Project/Datasets/Defect/validation/00-damage/0001.jpeg', target_size=(150, 150)) #
          x = img_to_array(img) # this is a Numpy array with shape (3, 256, 256)
          x = x.reshape((1,) + x.shape)/255 # this is a Numpy array with shape (1, 3, 256, 256)
          pred = model.predict(x)
          print (pred)
          pred_label = np.argmax(pred, axis=1)
          pred=int(pred*10)
          print (pred_label)
          d = {0: 'Minor', 1: 'Moderate', 2: 'Severe'}
          for key in d:
              if pred == key:
                  print ("Assessment: {} damage to vehicle".format(d[key]))
          print ("Severity assessment complete.")

1/1 [==============================] - 0s 39ms/step
[[0.29291642]]
[0]
Assessment: Severe damage to vehicle
Severity assessment complete.
```

Figure 11: Severity Assessment of the Car

6 Price Estimation Model

6.1 Importing Libraries

Before implementation of model, the required libraries needs to be imported for smooth execution. The below figure 12 shows the imported libraries of our research.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
from sklearn import metrics
```

Figure 12: Importing Libraries for Defect Detection Model

6.2 Loading and Reading the Data

The loading and reading the data is the first step to analyze the data by the following code shown in the figure 13.

```
In [67]: # Load data
         df = pd.read_csv('D:/Masters in DA/Research Project/Datasets/CAR DETAILS FROM CAR DEKHO.csv')

In [68]: #displaying information about the attributes present in the dataset
         df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 4340 entries, 0 to 4339
         Data columns (total 8 columns):
          #   Column         Non-Null Count  Dtype
         ---  ------         --------------  -----
          0   name           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
         dtypes: int64(3), object(5)
         memory usage: 271.4+ KB

In [69]: # show first few rows
         print(df.head(4))

                              name  year  selling_price  km_driven    fuel  \
         0           Maruti 800 AC  2007          60000      70000  Petrol
         1  Maruti Wagon R LXI Minor  2007         135000      50000  Petrol
         2      Hyundai Verna 1.6 SX  2012         600000     100000  Diesel
         3    Datsun RediGO T Option  2017         250000      46000  Petrol

           seller_type transmission        owner
         0  Individual       Manual  First Owner
         1  Individual       Manual  First Owner
         2  Individual       Manual  First Owner
```

Figure 13: Load and Read the Data

6.3 Data Preprocessing

The code shown in the below figure 4 describes the preprocessing stages in the project.

```
In [70]: #checking for null value
         df.isnull().sum()

Out[70]: name             0
         year             0
         selling_price    0
         km_driven        0
         fuel             0
         seller_type      0
         transmission     0
         owner            0
         dtype: int64

In [71]: #visualize the null data in a attributes using heatmap function
         import seaborn as mis
         mis.heatmap(df.isnull(), cmap='viridis')

Out[71]: <AxesSubplot:>
```

Figure 14: Checking the Null Values

```
In [72]: #Checking the NA values in dataframe
         df.dropna(axis=0,inplace=True)

In [73]: #Checking the duplicate values in dataframe
         df.duplicated().sum()

Out[73]: 763

In [74]: #Drop the duplicate data and save the data to other dataframe as data
         data = df.drop_duplicates()

In [75]: #Checking the duplicate values in dataframe
         data.duplicated().sum()

Out[75]: 0
```

Figure 15: Checking the Duplicate Values

6.4 Visualizing the features in Dataset

The visualization of data is effective to analyse the features in the dataset which describes the bar plots used in the project.

```
In [77]: plt.rcParams['figure.figsize'] = (8,4)
         sns.countplot(data['seller_type'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and pas
ing other arguments without an explicit keyword will result in an error or misinterpretation.
```

```
Out[77]: <AxesSubplot:xlabel='seller_type', ylabel='count'>
```

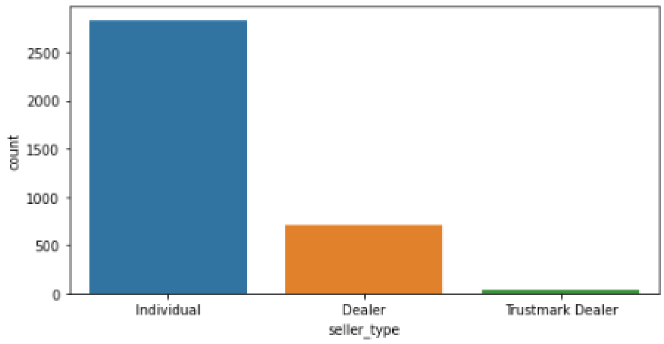


Figure 16: Visualization 1

```
In [85]: plt.rcParams['figure.figsize'] = (15,6)
         sns.countplot(data['year'])
         plt.title("year v/s vehicle's available for second's ")
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and pass
ing other arguments without an explicit keyword will result in an error or misinterpretation.
```

```
Out[85]: Text(0.5, 1.0, "year v/s vehicle's available for second's ")
```

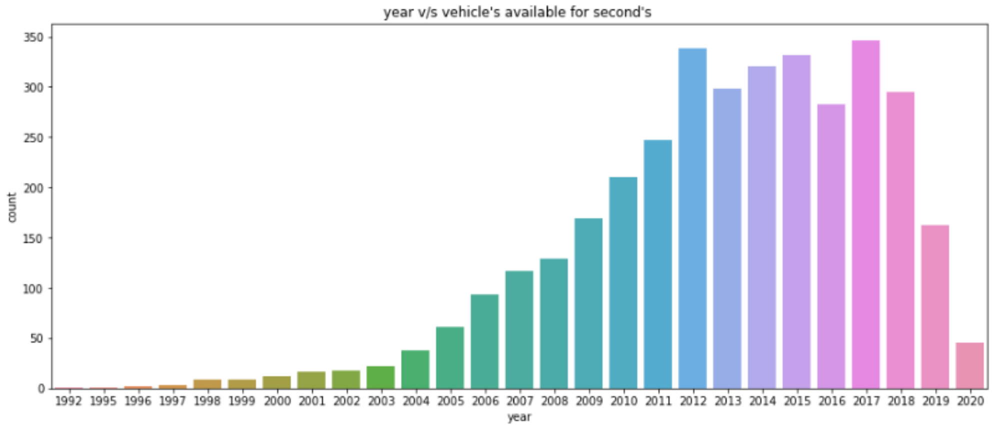


Figure 17: Visualization 2

6.5 Data Transformation

The data transformation is important step to change the data from one format to another which is done by the following code shown in the figure 18.

```
In [49]: data.replace({'fuel':{'Petrol':0,'Diesel':1,'LPG':2,'Electric':3, 'CNG':4}}, inplace=True)

C:\Users\HP\anaconda3\lib\site-packages\pandas\core\indexing.py:1676: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy

In [12]: data.replace({'seller_type':{'Dealer':0,'Individual':1, 'Trustmark Dealer':2}}, inplace=True)

In [13]: data.replace({'transmission':{'Manual':0,'Automatic':1}}, inplace=True)

In [14]: data.replace({'owner':{'First Owner':0,'Second Owner':1,'Third Owner':2,'Fourth & Above Owner':3,'Test Drive Car':4}}, inplace=Tı

In [15]: data.head()
```

Out[15]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	0	1	0	0
1	Maruti Wagon R LXI Minor	2007	135000	50000	0	1	0	0
2	Hyundai Verna 1.6 SX	2012	600000	100000	1	1	0	0
3	Datsun RediGO T Option	2017	250000	46000	0	1	0	0
4	Honda Amaze VX i-DTEC	2014	450000	141000	1	1	0	1

Figure 18: Data Augmentation and Normalization

6.6 Splitting of Train and Test Data

The dataset needs to be divided into training and validation data to develop the model as shown in the below figure 19

```
In [50]: #Spliting the data and Target
         X = data.drop(['name', 'selling_price'], axis=1)
         Y = data['selling_price']
```

Figure 19: Splitting of Train and Test Data

6.7 Linear Regression Model

The code below shows the model building and performance analysis of linear regression model was given in the figure 20.

```
In [52]: X_train,X_test, Y_train,Y_test = train_test_split(X,Y, test_size=0.3,random_state=2)

In [53]: #Training Data
         lin_reg_model = LinearRegression()
         lin_reg_model.fit(X_train,Y_train)

Out[53]: LinearRegression()

In [54]: training_data_prediction = lin_reg_model.predict(X_train)

In [59]: print("MSE value is : ",mean_squared_error(Y_train, training_data_prediction))
         print("R Square  value is : ",r2_score(Y_train, training_data_prediction))

         MSE value is :  154584053125.8305
         R Square  value is :  0.41735485636246517

In [60]: #Testing Data
         lin_reg_model = LinearRegression()
         lin_reg_model.fit(X_test,Y_test)

Out[60]: LinearRegression()

In [61]: testing_data_prediction = lin_reg_model.predict(X_test)

In [62]: print("MSE value is : ",mean_squared_error(Y_test, testing_data_prediction))
         print("R Square  value is : ",r2_score(Y_test, testing_data_prediction))

         MSE value is :  154341357657.48093
         R Square  value is :  0.3701376704976208
```

Figure 20: Linear Regression Model

6.8 Decision Tree Model

The code below shows the model building and performance analysis of decision tree model was given in the figure 21.

```
In [71]: #Training Data
         lin_reg_model = DecisionTreeRegressor()
         lin_reg_model.fit(X_train,Y_train)

Out[71]: DecisionTreeRegressor()

In [72]: training_data_prediction = lin_reg_model.predict(X_train)

In [73]: print("MSE value is : ",mean_squared_error(Y_train, training_data_prediction))
         print("R Square  value is : ",r2_score(Y_train, training_data_prediction))

         MSE value is :  13811136057.582731
         R Square  value is :  0.9479442336427961

In [63]: #Testing Data
         lin_reg_model = DecisionTreeRegressor()
         lin_reg_model.fit(X_test,Y_test)

Out[63]: DecisionTreeRegressor()

In [64]: testing_data_prediction = lin_reg_model.predict(X_test)

In [65]: print("MSE value is : ",mean_squared_error(Y_test, testing_data_prediction))
         print("R Square  value is : ",r2_score(Y_test, testing_data_prediction))

         MSE value is :  7749293908.295181
         R Square  value is :  0.9683753701071528
```

Figure 21: Decision Tree Model