PUBG PROJECT

Importing important libraries

```
In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
   ...: import seaborn as sns
```

Importing dataset:

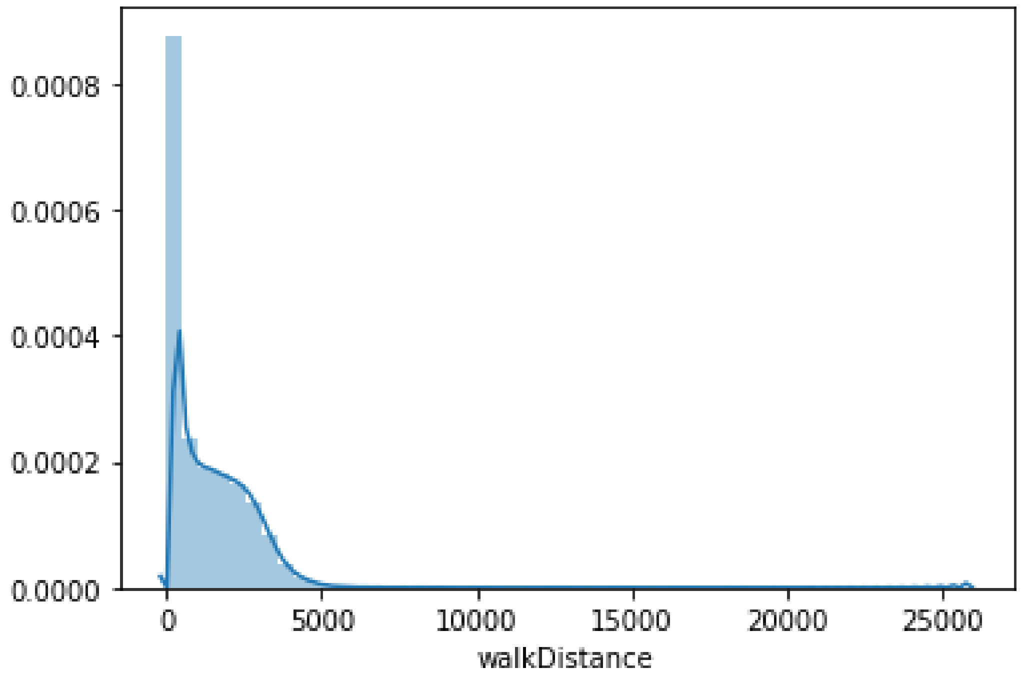
```
In [2]: dataset=pd.read_csv('pubg.csv')
```

Visualising dataset:

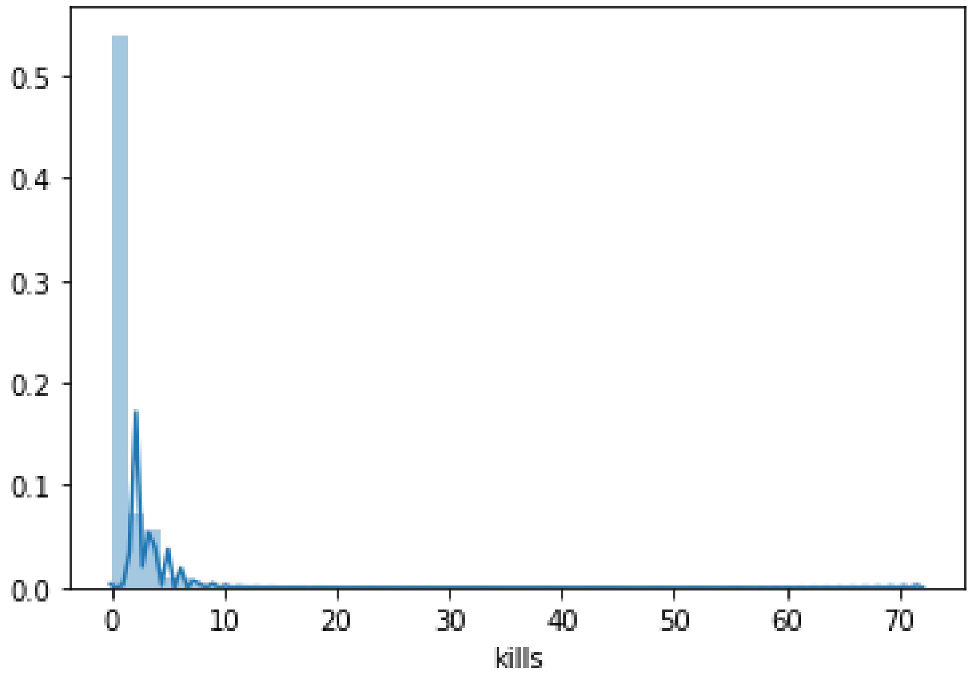
```
In [3]: dataset.head()
Out[3]:
Id groupId ... winPoints winPlacePerc
0 7f96b2f878858a 4d4b580de459be ... 1466 0.4444
1 eef90569b9d03c 684d5656442f9e ... 0 0.6400
2 1eaf90ac73de72 6a4a42c3245a74 ... 0 0.7755
3 4616d365dd2853 a930a9c79cd721 ... 0 0.1667
4 315c96c26c9aac de04010b3458dd ... 0 0.1875

[5 rows x 29 columns]

In [4]: dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4446966 entries, 0 to 4446965
Data columns (total 29 columns):
Id object
groupId object
matchId object
assists int64
boosts int64
damageDealt float64
DBNOs int64
headshotKills int64
heals int64
killPlace int64
killPoints int64
kills int64
killStreaks int64
longestKill float64
matchDuration int64
matchType object
maxPlace int64
numGroups int64
rankPoints int64
revives int64
rideDistance float64
roadKills int64
swimDistance float64
teamKills int64
vehicleDestroys int64
```

```
walkDistance float64
weaponsAcquired int64
winPoints int64
winPlacePerc float64
dtypes: float64(6), int64(19), object(4)
memory usage: 983.9+ MB
```

```
In [5]: sns.distplot(dataset['walkDistance'])
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8ac20278>
```

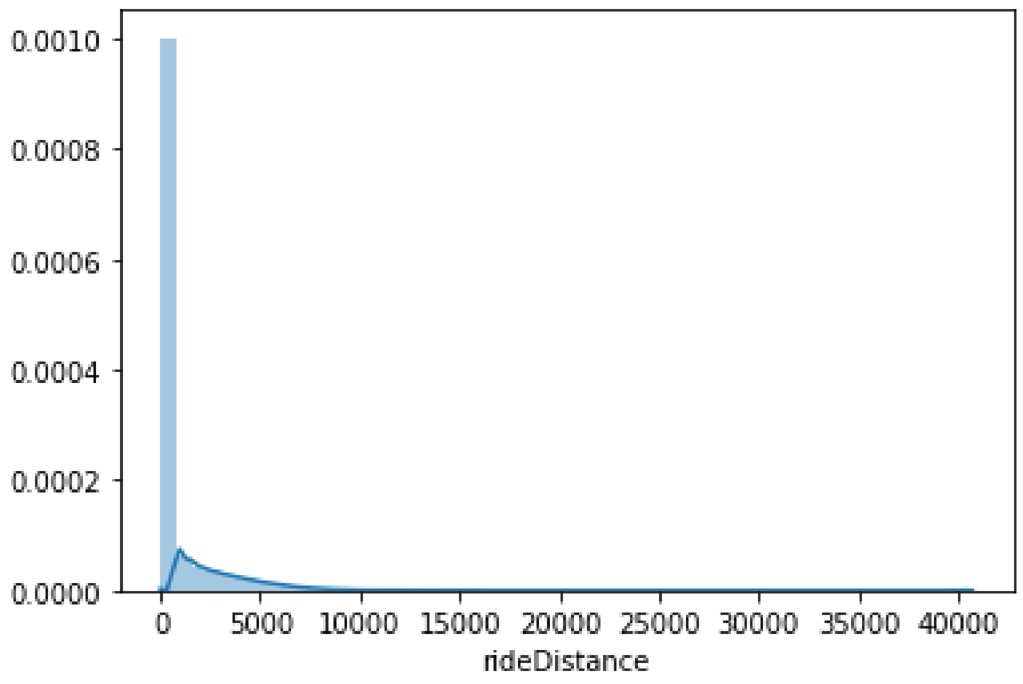


```
In [6]: sns.distplot(dataset['kills'])
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8ad74e48>
```

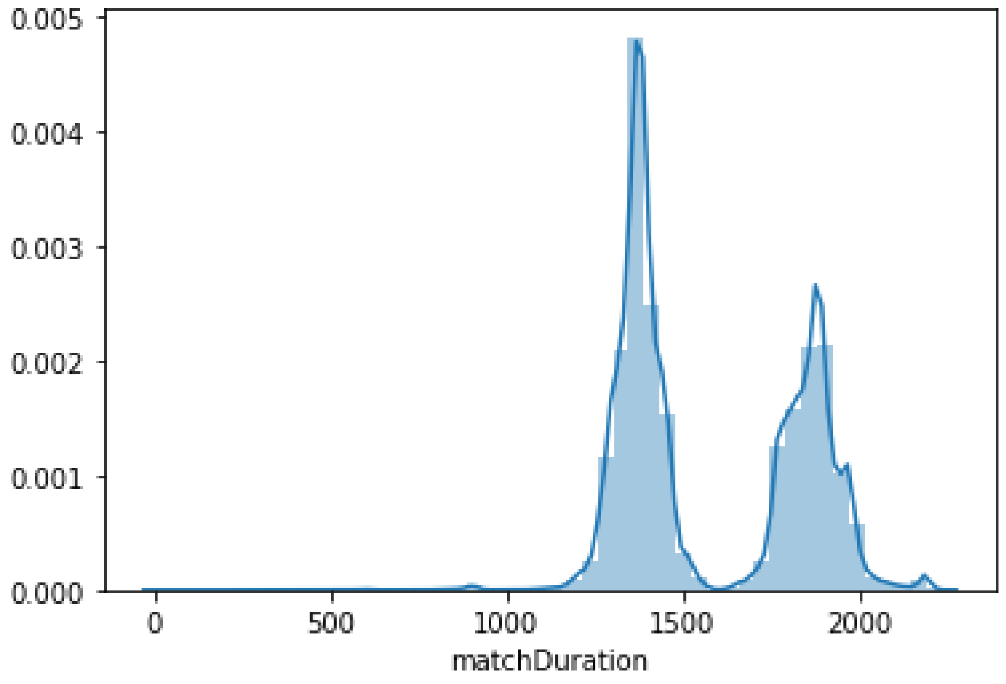


```
In [7]: sns.distplot(dataset['rideDistance'])
```

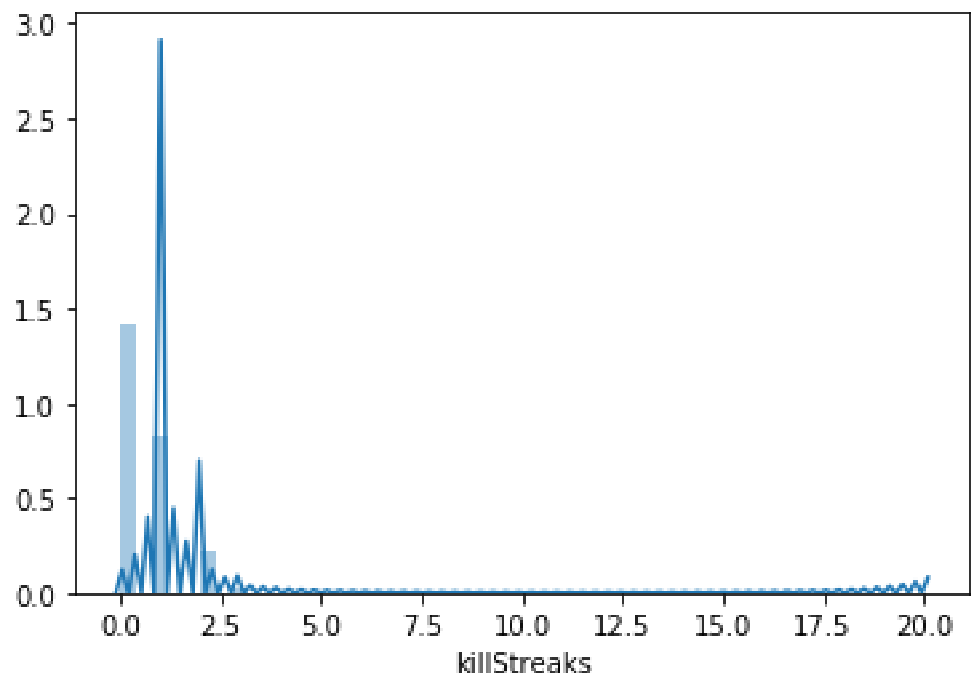
```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8ae9c048>
```



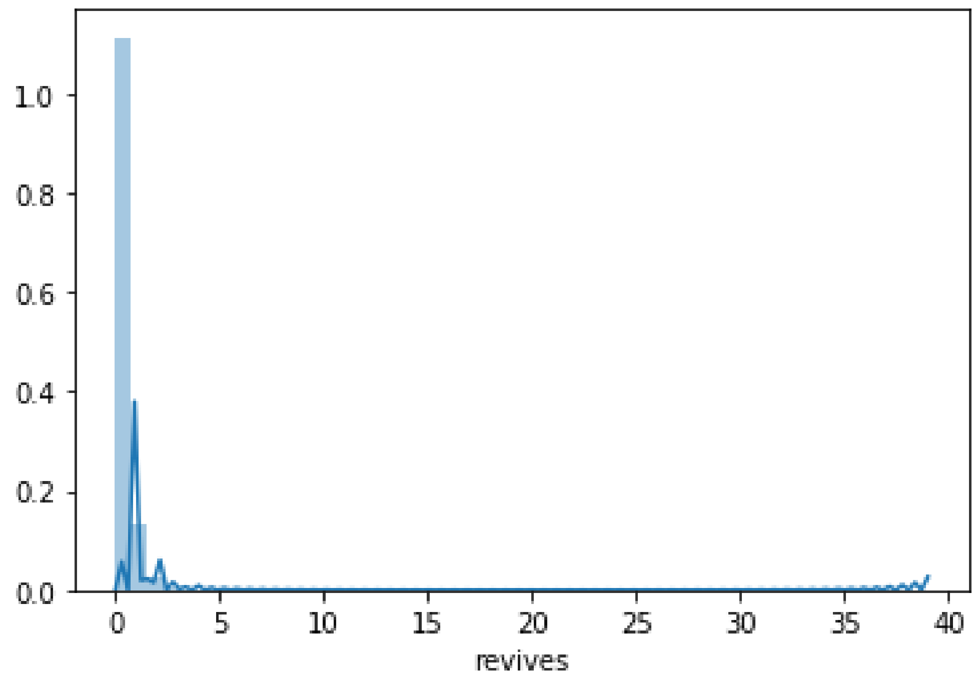
```
In [8]: sns.distplot(dataset['matchDuration'])
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8ae9c128>
```



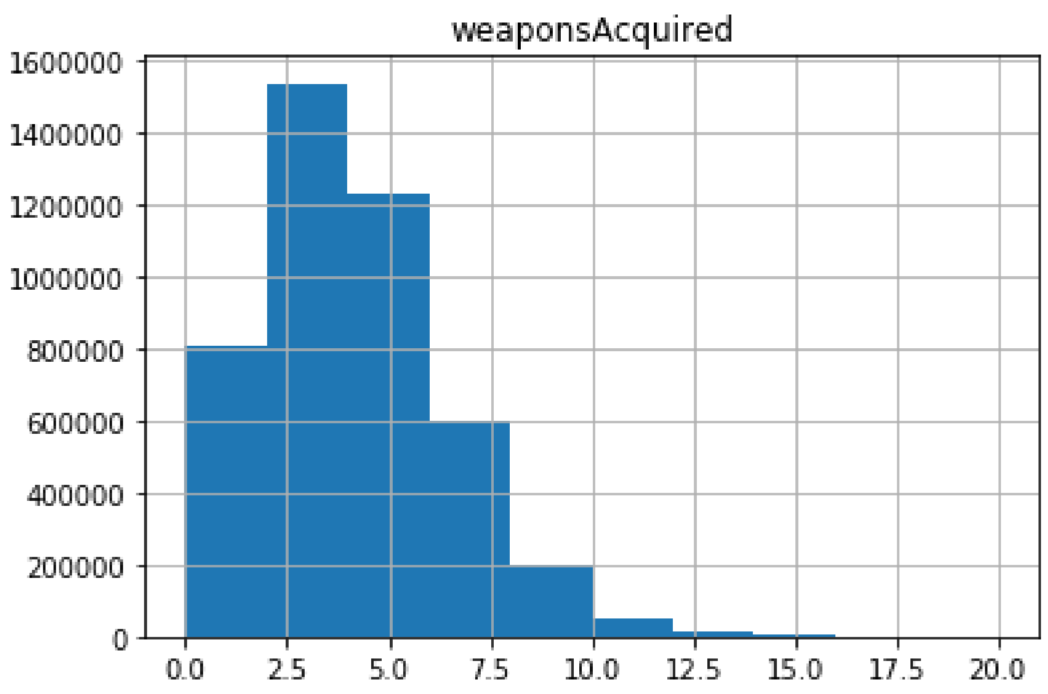
```
In [9]: sns.distplot(dataset['killStreaks'])
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8afa9710>
```



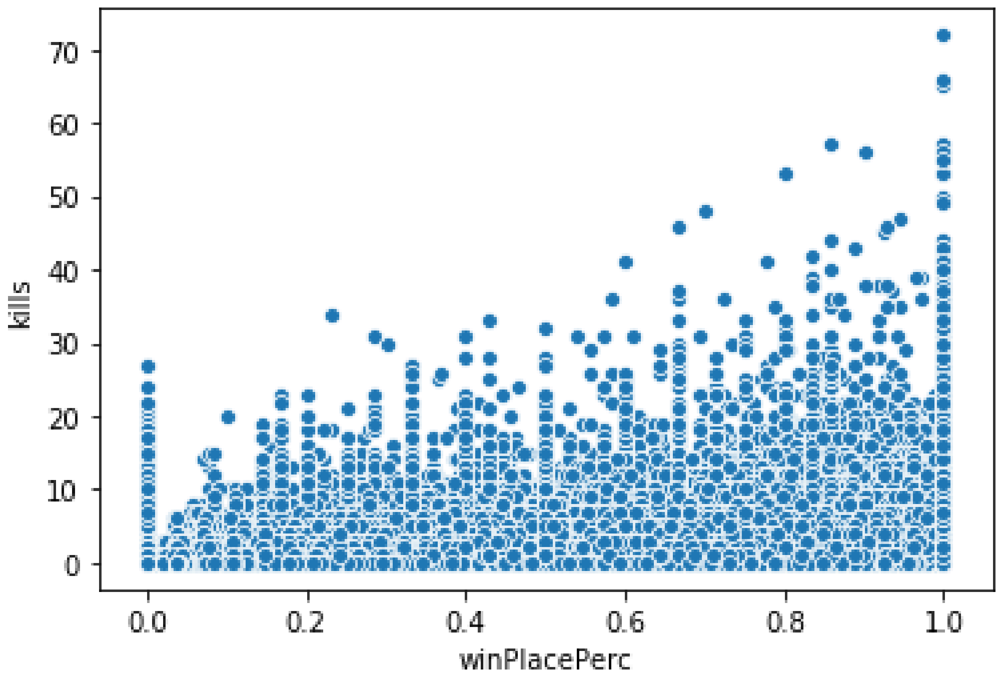
```
In [10]: sns.distplot(dataset['revives'])
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8b170cf8>
```



```
In [11]: dataset.hist('weaponsAcquired',range=(0,20))
Out[11]:
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028D8B21BF28>]],
dtype=object)
```



```
In [12]: sns.scatterplot(x=dataset['winPlacePerc'],y=dataset["kills"])
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x28d8b2965f8>
```



Removal of undesired data :

Highest kill record is 39. So, player with more than 39 kills might be hacker

```
In [13]: dataset.drop(dataset[dataset['kills']>39].index,inplace=True)
```

```
In [14]: dataset.drop(dataset[dataset['rideDistance']>15000].index,inplace=True)
In [15]:
dataset['totaldist']=dataset['rideDistance']+dataset['walkDistance']+dataset['swim
Distance']

In [16]:
dataset['killwithoutmoving']=((dataset['kills']>0)&(dataset['totaldist']==0))
    ...:
dataset.drop(dataset[dataset['killwithoutmoving']==True].index,inplace=True)
```

Splitting of independent and dependent parameters:

```
In [17]: X=dataset.iloc[:,3:28].values
    ...: y=dataset.iloc[:,28].values
```

Encoding using onehotencoder:

```
In [18]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
    ...: labelencoder_X_1 = LabelEncoder()
    ...: X[:, 12] = labelencoder_X_1.fit_transform(X[:, 12])
    ...: onehotencoder = OneHotEncoder(categorical_features = [12])
    ...: X = onehotencoder.fit_transform(X).toarray()
    ...: X = X[:, 1:] #for overcome the onehotencoder trap
C:\Users\SURYANSH\Anaconda3\lib\site-
packages\sklearn\preprocessing\_encoders.py:414: FutureWarning: The handling of
integer data will change in version 0.22. Currently, the categories are determined
based on the range [0, max(values)], while in the future they will be determined
based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the
categories to integers, then you can now use the OneHotEncoder directly.
warnings.warn(msg, FutureWarning)
C:\Users\SURYANSH\Anaconda3\lib\site-
packages\sklearn\preprocessing\_encoders.py:450: DeprecationWarning: The
'categorical_features' keyword is deprecated in version 0.20 and will be removed
in 0.22. You can use the ColumnTransformer instead.
"use the ColumnTransformer instead.", DeprecationWarning)
```

Splitting of traing and test data:

```
In [19]: from sklearn.model_selection import train_test_split
    ...: X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20,random_state=0)
```

Scaling:

```
In [20]: from sklearn.preprocessing import StandardScaler
    ...: sc = StandardScaler()
    ...: X_train = sc.fit_transform(X_train)
    ...: X_test = sc.transform(X_test)
```

Importing keras:

```
In [21]: import keras
    ...: from keras.models import Sequential
    ...: from keras.layers import Dense
Using TensorFlow backend.
```

Initialising ANN Model

```
In [22]: classifier = Sequential()

In [23]: classifier.add(Dense(output_dim = 80, init = 'uniform', activation =
'relu', input_dim = 39))
    ...: # Adding the second hidden layer
    ...: classifier.add(Dense(output_dim = 80, init = 'uniform', activation =
'relu'))
    ...: # Adding the third hidden layer
    ...: classifier.add(Dense(output_dim =80, init = 'uniform', activation =
'relu'))
    ...: # Adding the output layer
    ...: classifier.add(Dense(output_dim = 1, init = 'uniform', activation =
'linear'))
    ...: # Compiling the ANN
    ...: classifier.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])
__main__:1: UserWarning: Update your `Dense` call to the Keras 2 API:
`Dense(activation="relu", input_dim=39, units=80, kernel_initializer="uniform")`
__main__:3: UserWarning: Update your `Dense` call to the Keras 2 API:
`Dense(activation="relu", units=80, kernel_initializer="uniform")`
__main__:5: UserWarning: Update your `Dense` call to the Keras 2 API:
`Dense(activation="relu", units=80, kernel_initializer="uniform")`
__main__:7: UserWarning: Update your `Dense` call to the Keras 2 API:
`Dense(activation="linear", units=1, kernel_initializer="uniform")`
```

Training of Model:

```
In [24]: history=classifier.fit(X_train, y_train, batch_size = 254, nb_epoch =
10,validation_split=0.20)
__main__:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed
`epochs`.
Train on 2844734 samples, validate on 711184 samples
Epoch 1/10
2844734/2844734 [==============================] - 33s 12us/step - loss: 0.0080 -
mean_absolute_error: 0.0627 - val_loss: 0.0073 - val_mean_absolute_error: 0.0608
Epoch 2/10
2844734/2844734 [==============================] - 29s 10us/step - loss: 0.0068 -
mean_absolute_error: 0.0589 - val_loss: 0.0066 - val_mean_absolute_error: 0.0583
Epoch 3/10
2844734/2844734 [==============================] - 30s 11us/step - loss: 0.0066 -
mean_absolute_error: 0.0581 - val_loss: 0.0065 - val_mean_absolute_error: 0.0575
Epoch 4/10
2844734/2844734 [==============================] - 33s 12us/step - loss: 0.0065 -
mean_absolute_error: 0.0577 - val_loss: 0.0064 - val_mean_absolute_error: 0.0573
Epoch 5/10
```

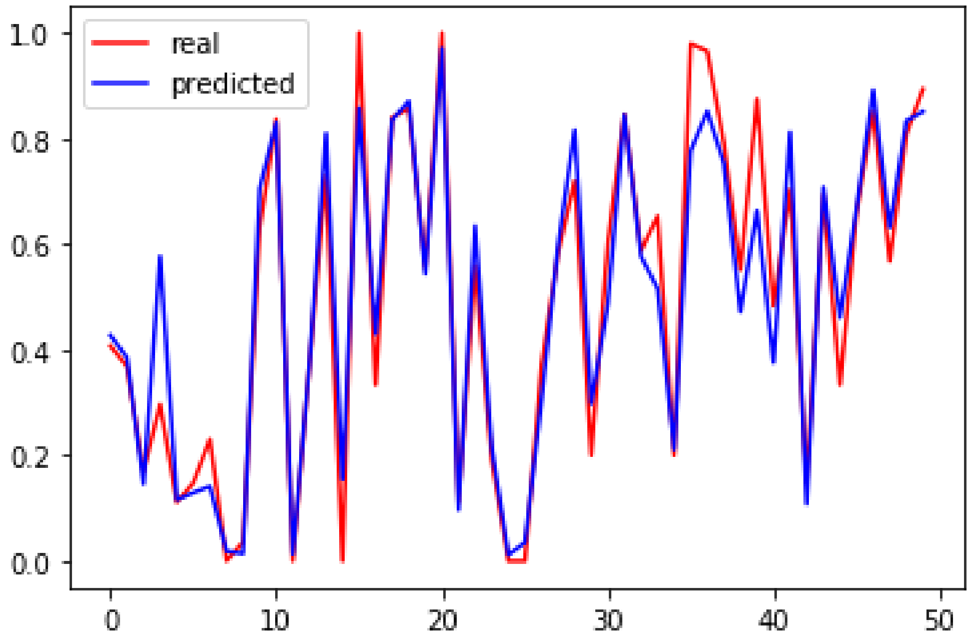
```
2844734/2844734 [==============================] - 34s 12us/step - loss: 0.0065 -
mean_absolute_error: 0.0575 - val_loss: 0.0065 - val_mean_absolute_error: 0.0577
Epoch 6/10
2844734/2844734 [==============================] - 35s 12us/step - loss: 0.0064 -
mean_absolute_error: 0.0574 - val_loss: 0.0064 - val_mean_absolute_error: 0.0573
Epoch 7/10
2844734/2844734 [==============================] - 32s 11us/step - loss: 0.0064 -
mean_absolute_error: 0.0573 - val_loss: 0.0065 - val_mean_absolute_error: 0.0576
Epoch 8/10
2844734/2844734 [==============================] - 32s 11us/step - loss: 0.0064 -
mean_absolute_error: 0.0572 - val_loss: 0.0064 - val_mean_absolute_error: 0.0572
Epoch 9/10
2844734/2844734 [==============================] - 31s 11us/step - loss: 0.0064 -
mean_absolute_error: 0.0571 - val_loss: 0.0065 - val_mean_absolute_error: 0.0571
Epoch 10/10
2844734/2844734 [==============================] - 38s 13us/step - loss: 0.0064 -
mean_absolute_error: 0.0570 - val_loss: 0.0065 - val_mean_absolute_error: 0.0574
```

Prediction:

```
In [25]: prediction=classifier.predict(X_test)
```

Visualisation of predicted and real data:

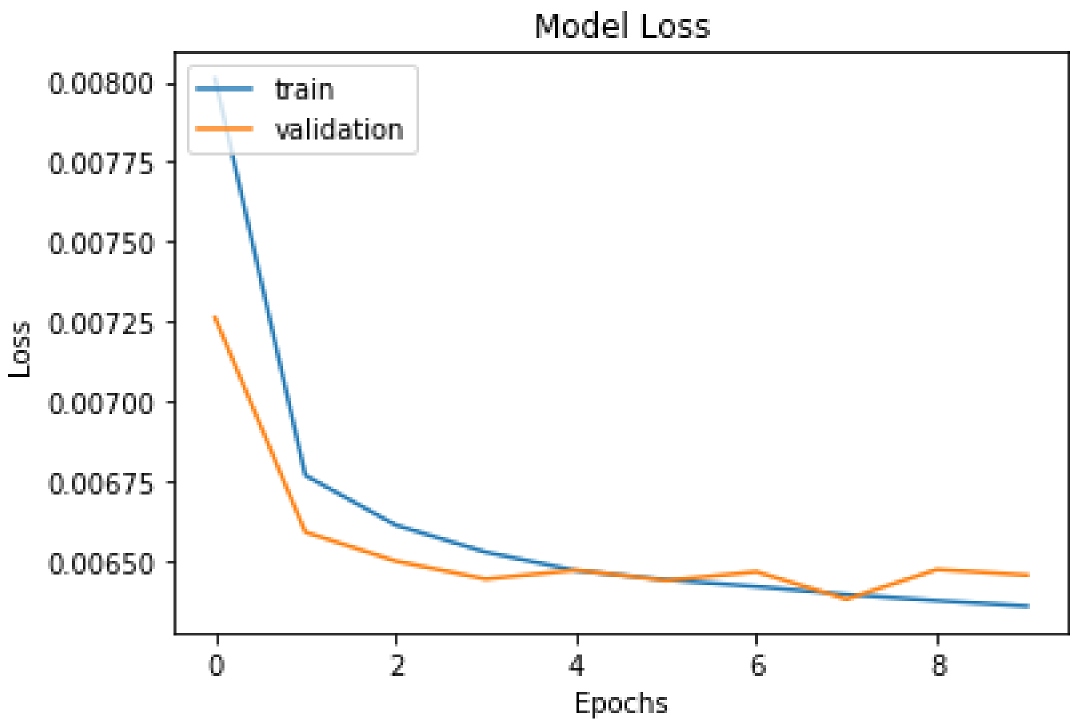
```
In [27]: plt.plot(y_test[100:150],color='red',label="real")
    ...: plt.plot(prediction[100:150],color='blue',label="predicted")
    ...: plt.legend()
    ...: plt.show()
```



Visualisation of epochs Vs loss:

```
In [28]: plt.plot(history.history['loss'])
```

```
   ...: plt.plot(history.history['val_loss'])
   ...: plt.title('Model Loss')
   ...: plt.ylabel('Loss')
   ...: plt.xlabel("Epochs")
   ...: plt.legend(['train','validation'],loc='upper left')
   ...: plt.show()
```



```
In [29]:
```