Linear Regression.

We will start by importing the required libraries.

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing the data.

In [ ]:

```
tweets = pd.read_csv('https://raw.githubusercontent.com/sandeeptuluri/Machine-Learning/ma
in/Twitter.csv')
```

In [ ]:

```
tweets.head(2)
```

Out[ ]:

	Unnamed: 0	Id	Post Contet	Sentiment score	Post Length	Hashtag count	Content URL count	Twee coun
0	41370	6d967b125fcecba6357dbc43f8f380cf2d6d7a51	Sana all na lang.	0.0	17.0	0.0	0.0	1660.
1	27955	22dc5f808a8589186767412f39e5c88ae9753d04	キスマイ玉森裕太「ボス恋」台本の裏話明かす \n\n@TBS_asachan @bosskoi...	19.3	84.0	0.0	1.0	318924.

In [ ]:

```
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             50000 non-null  int64
 1   Id                     50000 non-null  object
 2   Post Contet            50000 non-null  object
 3   Sentiment score        50000 non-null  float64
 4   Post Length            50000 non-null  float64
 5   Hashtag count          50000 non-null  float64
 6   Content URL count      50000 non-null  float64
 7   Tweet count            50000 non-null  float64
 8   Followers count        50000 non-null  float64
 9   Listed Count           50000 non-null  int64
 10  Media Type             50000 non-null  object
 11  Published DateTime     50000 non-null  object
 12  Mentions Count         50000 non-null  float64
 13  Post author verified   50000 non-null  float64
 14  Likes                  50000 non-null  float64
 15  Shares                 50000 non-null  float64
 16  Comments               50000 non-null  float64
 17  Impact                 50000 non-null  float64
dtypes: float64(12), int64(2), object(4)
memory usage: 6.9+ MB
```

The data looks clean and clear.

In [ ]:

```
tweets.describe()
```

Out[ ]:

	Unnamed: 0	Sentiment score	Post Length	Hashtag count	Content URL count	Tweet count	Followers count	Listed Count
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	5.000000e+04	5.000000e+04	50000.000000
mean	14193.578860	1.068916	154.692360	0.687520	0.480260	2.414257e+05	4.648759e+06	10069.683200
std	10363.500433	10.436746	79.099411	1.346979	0.526019	1.607467e+06	1.254513e+07	28384.958681
min	0.000000	-20.000000	1.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	6017.000000	0.000000	94.000000	0.000000	0.000000	1.123775e+04	1.053900e+04	2.000000
50%	12076.500000	0.000000	142.000000	0.000000	0.000000	5.273800e+04	3.551225e+05	555.500000
75%	20650.250000	0.000000	215.000000	1.000000	1.000000	2.595015e+05	2.809978e+06	6171.000000
max	43879.000000	20.000000	373.000000	21.000000	7.000000	5.044408e+07	1.144406e+08	568139.000000

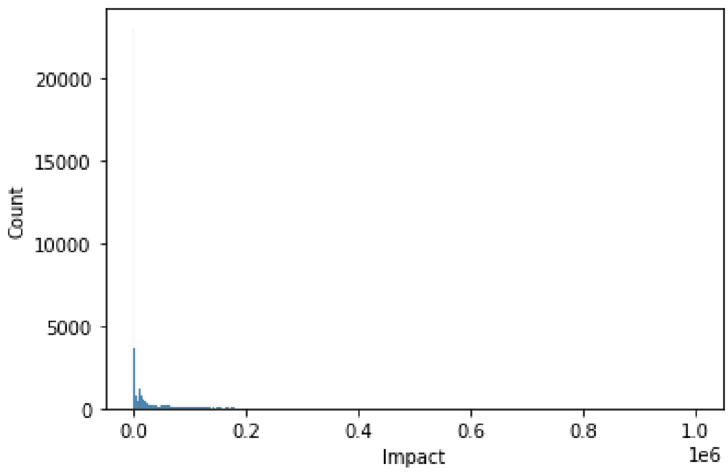
Exploratory Data Analysis.

In [ ]:

```
sns.histplot(tweets.Impact)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f94374dd0>
```



It seems like very few number of tweets got high impact.

In [ ]:

```
tweets.Impact.max()
```

Out[ ]:

```
997980.0
```

In [ ]:

```
tweets[tweets.Impact == 997980]['Post Contet'].iloc[0]
```

Out[ ]:

```
'PS5 Global launch schedule: https://t.co/zgwfUX6iVl'
```

The above tweet got the highest impact from the overall tweets.

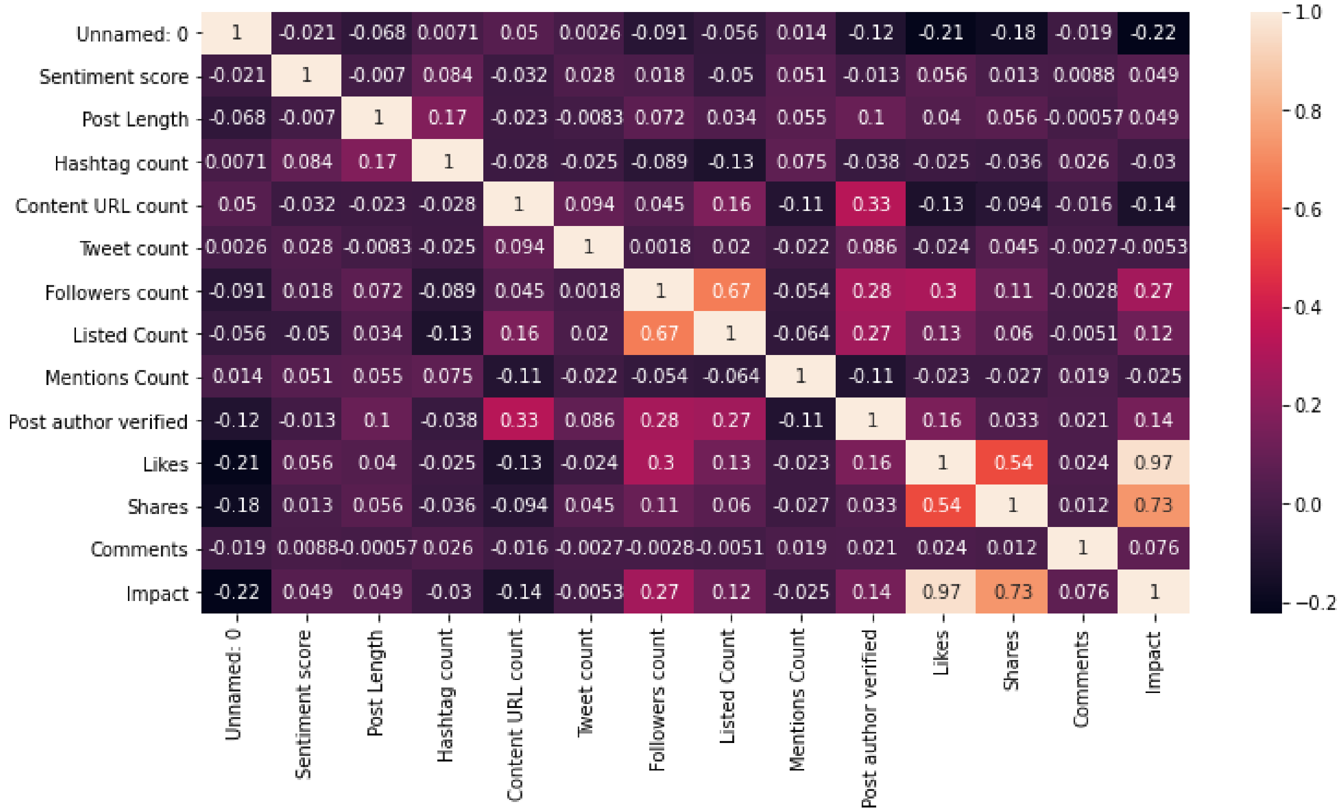
Let's check the correlation between in the dataset.

In [ ]:

```
plt.figure(figsize=(12,6))
sns.heatmap(tweets.corr(), annot=True)
```

Out[ ]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f5f95fb2090&gt;
```



From the dataset the most correlated features are 'Content URL count', 'Post author verified', 'Followers Count', 'listed count', 'Likes', 'Impact', 'Shares'. so we will consider these features for the training the model.

Linear Regression Model.

Train Test Split.

In [ ]:

```
from sklearn.model_selection import train_test_split
```

In [ ]:

```
X = tweets[['Content URL count','Followers count','Listed Count','Post author verified',
'Likes', 'Shares']]
y = tweets['Impact']
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10
1)
```

In [ ]:

```
from sklearn.linear_model import LinearRegression
```

Loading the model and fitting the train data.

In [ ]:

```
linreg = LinearRegression()
```

In [ ]:

```
model = linreg.fit(X_train,y_train)
```

In [ ]:

```
from sklearn import metrics

def print_metrics(a, b):
  mae = metrics.mean_absolute_error(a, b)
  mse = metrics.mean_squared_error(a, b)
  rmse = np.sqrt(metrics.mean_squared_error(a,b))
  r_squared = metrics.r2_score(a, b)
  print("MAE", mae)
  print("MSE", mse)
  print("RMSE", rmse)
  print("R2_squared", r_squared)
  print('--------------------------------------')
```

Printing the metrics.

In [ ]:

```
train_pred = model.predict(X_train)
test_pred = model.predict(X_test)
```

In [ ]:

```
print('Training data evaluation:\n----------------------------')
print_metrics(y_train, train_pred)
print('Test data evaluation:\n----------------------------')
print_metrics(y_test, test_pred)
```

```
Training data evaluation:
----------------------------
MAE 390.49829714982894
MSE 39866179.92028137
RMSE 6313.967050934093
R2_squared 0.9963284216068522
--------------------------------------
Test data evaluation:
----------------------------
MAE 311.18492577240653
MSE 5359971.045152342
RMSE 2315.1611272549353
R2_squared 0.9994681673873835
--------------------------------------
```

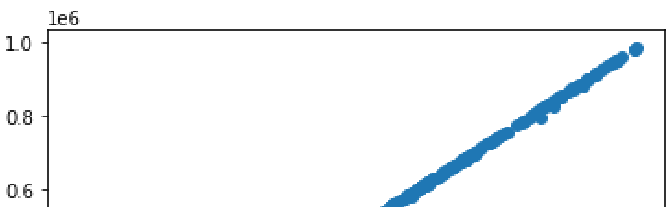
An R2_squared score of 1.0 indicates that the data perfectly fits the model. here, as our model's R2_squared score of 0.9994 indicates that our data perfectly fits our model.

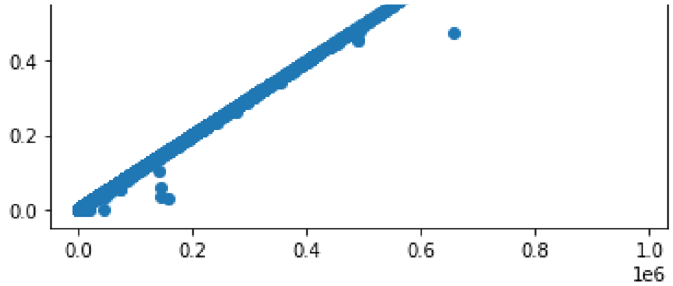
In [ ]:

```
plt.scatter(y_test, test_pred)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7f5f837ed350>
```





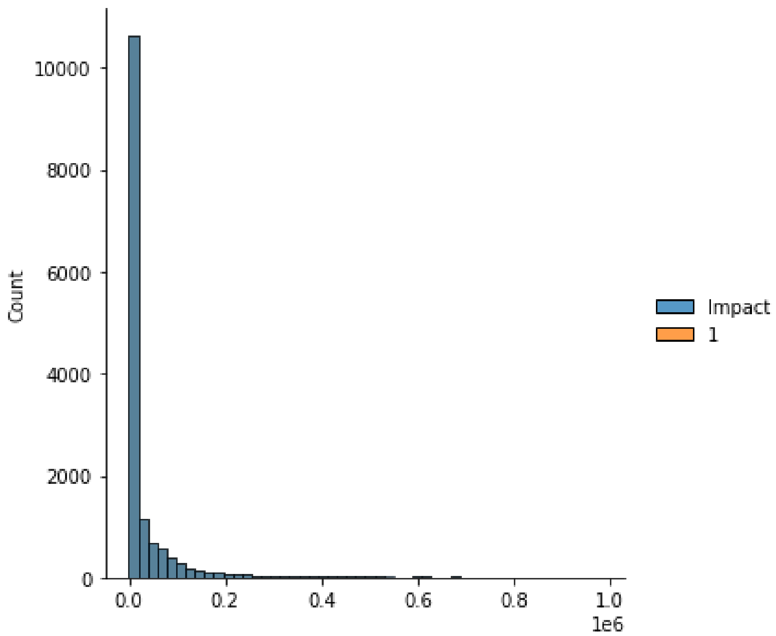
Scatter plot also shows the perfect graph and it is highly linear, indicating the best accuracy and best model.

In [ ]:

```
sns.displot((y_test, test_pred), bins = 50)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7f5f8a18d090>
```



In [ ]:

```
print(model.intercept_)
```

```
76.8176836028506
```

In [ ]:

```
coeff_df = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[ ]:

	Coefficient
Content URL count	-287.280174
Followers count	-0.000006
Listed Count	-0.000829
Post author verified	399.638523
Likes	10.009965
Shares	10.004375

The coefficient indicates how much the dependent variable increases (if positive), and decreases (if negative), if the independent variable increases by one. if the post author verified, likes, shares increases by one, the impact of the tweets increases by (399,10,10) times.

DECISION TREE MODEL.

Importing the model and other required libraries.

In [ ]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [ ]:

```
sns.set_style('whitegrid')
```

Training the model.

In [ ]:

```
dtm = DecisionTreeRegressor(max_depth=5, min_samples_split=5, max_leaf_nodes=10)
```

In [ ]:

```
dtm.fit(X_train,y_train)
```

Out[ ]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=5,
                      max_features=None, max_leaf_nodes=10,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=5,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

Predicting the model scores.

In [ ]:

```
train_dtm_pred = dtm.predict(X_train)
test_dtm_pred = dtm.predict(X_test)
```

In [ ]:

```
print('Training data evaluation:\n----------------------------')
print_metrics(y_train, train_dtm_pred)
print('Test data evaluation:\n----------------------------')
print_metrics(y_test, test_dtm_pred)
```

```
Training data evaluation:
----------------------------
MAE 9297.597976476207
MSE 543755944.42697
RMSE 23318.575094267017
R2_squared 0.9499213975179979
--------------------------------------
Test data evaluation:
----------------------------
MAE 9125.75120292198
MSE 446548036.65582645
RMSE 21131.68324236918
R2_squared 0.9556921470297507
--------------------------------------
```

It seems like, data pretty much fits the model, as we are having good R2 score in both data sets.

RANDOM FOREST REGRESSOR.

Importing the required libraries.

In [ ]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:

```
rf = RandomForestRegressor(n_estimators=1000)
```

In [ ]:

```
rf.fit(X_train,y_train)
```

Out[ ]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [ ]:

```
train_rf_pred = rf.predict(X_train)
test_rf_pred = rf.predict(X_test)
```

In [ ]:

```
print('Training data evaluation:\n----------------------------')
print_metrics(y_train, train_rf_pred)
print('Test data evaluation:\n----------------------------')
print_metrics(y_test, test_rf_pred)
```

```
Training data evaluation:
----------------------------
MAE 214.1747494558196
MSE 4507666.964254432
RMSE 2123.126695290329
R2_squared 0.9995848548152204
--------------------------------------
Test data evaluation:
----------------------------
MAE 489.73985983942964
MSE 13605186.139588151
RMSE 3688.520860668698
R2_squared 0.99865005209379
--------------------------------------
```

Random Forest Regressor gives the good results of test data as R2_score is 0.998, which means it is a good fit for the model.

Artificial Neural Networks

Importing the required libraries.

In [ ]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

In [ ]:

```
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

In [ ]:

```
model = Sequential()

model.add(Dense(6,activation='relu'))

model.add(Dense(32,activation='relu'))

model.add(Dense(64,activation='relu'))

model.add(Dense(128,activation='relu'))

model.add(Dense(512,activation='relu'))

model.add(Dropout(0.1))

model.add(Dense(1))

model.compile(optimizer=Adam(learning_rate=0.0005),loss='mse')
```

In [ ]:

```
model.fit(X_train, y_train, validation_data=(X_test,y_test),batch_size=256, epochs=200)
```

```
Epoch 1/200
137/137 [==============================] - 2s 10ms/step - loss: 219152048.0000 - val_loss
: 9390925.0000
Epoch 2/200
137/137 [==============================] - 1s 8ms/step - loss: 121870304.0000 - val_loss:
14394918.0000
Epoch 3/200
137/137 [==============================] - 1s 9ms/step - loss: 122927328.0000 - val_loss:
516842048.0000
Epoch 4/200
137/137 [==============================] - 1s 9ms/step - loss: 469401344.0000 - val_loss:
2556896256.0000
Epoch 5/200
137/137 [==============================] - 1s 8ms/step - loss: 686001088.0000 - val_loss:
14003909.0000
Epoch 6/200
137/137 [==============================] - 1s 8ms/step - loss: 60609724.0000 - val_loss:
9165858.0000
Epoch 7/200
137/137 [==============================] - 1s 8ms/step - loss: 58404716.0000 - val_loss:
11046266.0000
Epoch 8/200
137/137 [==============================] - 1s 9ms/step - loss: 60447932.0000 - val_loss:
47953696.0000
Epoch 9/200
137/137 [==============================] - 1s 9ms/step - loss: 81789688.0000 - val_loss:
```

```
7623756.5000
Epoch 10/200
137/137 [==============================] - 1s 9ms/step - loss: 119023144.0000 - val_loss:
23115070.0000
Epoch 11/200
137/137 [==============================] - 1s 9ms/step - loss: 112382824.0000 - val_loss:
78692472.0000
Epoch 12/200
137/137 [==============================] - 1s 9ms/step - loss: 136819152.0000 - val_loss:
39795232.0000
Epoch 13/200
137/137 [==============================] - 1s 9ms/step - loss: 74963376.0000 - val_loss:
32829348.0000
Epoch 14/200
137/137 [==============================] - 1s 9ms/step - loss: 78529096.0000 - val_loss:
10347565.0000
Epoch 15/200
137/137 [==============================] - 1s 9ms/step - loss: 82785416.0000 - val_loss:
11541900.0000
Epoch 16/200
137/137 [==============================] - 1s 8ms/step - loss: 107471400.0000 - val_loss:
26852114.0000
Epoch 17/200
137/137 [==============================] - 1s 9ms/step - loss: 971108672.0000 - val_loss:
71485992.0000
Epoch 18/200
137/137 [==============================] - 1s 8ms/step - loss: 68729848.0000 - val_loss:
12194340.0000
Epoch 19/200
137/137 [==============================] - 1s 9ms/step - loss: 64602084.0000 - val_loss:
9500862.0000
Epoch 20/200
137/137 [==============================] - 1s 8ms/step - loss: 79025656.0000 - val_loss:
62244724.0000
Epoch 21/200
137/137 [==============================] - 1s 9ms/step - loss: 73610840.0000 - val_loss:
7795760.0000
Epoch 22/200
137/137 [==============================] - 1s 9ms/step - loss: 64056220.0000 - val_loss:
8008179.0000
Epoch 23/200
137/137 [==============================] - 1s 8ms/step - loss: 67329816.0000 - val_loss:
13513824.0000
Epoch 24/200
137/137 [==============================] - 1s 9ms/step - loss: 280740352.0000 - val_loss:
99260920.0000
Epoch 25/200
137/137 [==============================] - 1s 9ms/step - loss: 77527656.0000 - val_loss:
8913170.0000
Epoch 26/200
137/137 [==============================] - 1s 9ms/step - loss: 71467952.0000 - val_loss:
10941984.0000
Epoch 27/200
137/137 [==============================] - 1s 8ms/step - loss: 77277848.0000 - val_loss:
235335488.0000
Epoch 28/200
137/137 [==============================] - 1s 9ms/step - loss: 162750656.0000 - val_loss:
37892408.0000
Epoch 29/200
137/137 [==============================] - 1s 9ms/step - loss: 98345608.0000 - val_loss:
160033984.0000
Epoch 30/200
137/137 [==============================] - 1s 9ms/step - loss: 75992024.0000 - val_loss:
11249326.0000
Epoch 31/200
137/137 [==============================] - 1s 8ms/step - loss: 91084176.0000 - val_loss:
14542670.0000
Epoch 32/200
137/137 [==============================] - 1s 8ms/step - loss: 80011760.0000 - val_loss:
17025392.0000
Epoch 33/200
137/137 [==============================] - 1s 9ms/step - loss: 218999888.0000 - val_loss:
```

```
151111424.0000
Epoch 34/200
137/137 [==============================] - 1s 9ms/step - loss: 73319224.0000 - val_loss:
8415074.0000
Epoch 35/200
137/137 [==============================] - 1s 10ms/step - loss: 57739612.0000 - val_loss:
24428410.0000
Epoch 36/200
137/137 [==============================] - 1s 9ms/step - loss: 151134384.0000 - val_loss:
49030556.0000
Epoch 37/200
137/137 [==============================] - 1s 9ms/step - loss: 304539680.0000 - val_loss:
23785216.0000
Epoch 38/200
137/137 [==============================] - 1s 8ms/step - loss: 72280936.0000 - val_loss:
9528635.0000
Epoch 39/200
137/137 [==============================] - 1s 9ms/step - loss: 62077536.0000 - val_loss:
10141555.0000
Epoch 40/200
137/137 [==============================] - 1s 9ms/step - loss: 72615416.0000 - val_loss:
9219355.0000
Epoch 41/200
137/137 [==============================] - 1s 8ms/step - loss: 62384228.0000 - val_loss:
8970343.0000
Epoch 42/200
137/137 [==============================] - 1s 9ms/step - loss: 63094608.0000 - val_loss:
15213036.0000
Epoch 43/200
137/137 [==============================] - 1s 9ms/step - loss: 101307560.0000 - val_loss:
14744996.0000
Epoch 44/200
137/137 [==============================] - 1s 9ms/step - loss: 159416768.0000 - val_loss:
78124808.0000
Epoch 45/200
137/137 [==============================] - 1s 8ms/step - loss: 86817880.0000 - val_loss:
8512527.0000
Epoch 46/200
137/137 [==============================] - 1s 9ms/step - loss: 68293440.0000 - val_loss:
10105258.0000
Epoch 47/200
137/137 [==============================] - 1s 9ms/step - loss: 61421248.0000 - val_loss:
24835466.0000
Epoch 48/200
137/137 [==============================] - 1s 8ms/step - loss: 85730616.0000 - val_loss:
16304793.0000
Epoch 49/200
137/137 [==============================] - 1s 8ms/step - loss: 68819736.0000 - val_loss:
13785740.0000
Epoch 50/200
137/137 [==============================] - 1s 8ms/step - loss: 105270920.0000 - val_loss:
11159060.0000
Epoch 51/200
137/137 [==============================] - 1s 8ms/step - loss: 57444308.0000 - val_loss:
13209631.0000
Epoch 52/200
137/137 [==============================] - 1s 9ms/step - loss: 60881956.0000 - val_loss:
13257614.0000
Epoch 53/200
137/137 [==============================] - 1s 9ms/step - loss: 65639488.0000 - val_loss:
177897712.0000
Epoch 54/200
137/137 [==============================] - 1s 9ms/step - loss: 199417968.0000 - val_loss:
14264597.0000
Epoch 55/200
137/137 [==============================] - 1s 8ms/step - loss: 60458504.0000 - val_loss:
18150846.0000
Epoch 56/200
137/137 [==============================] - 1s 8ms/step - loss: 59106792.0000 - val_loss:
34036464.0000
Epoch 57/200
137/137 [==============================] - 1s 8ms/step - loss: 104185352.0000 - val_loss:
```

```
301039776.0000
Epoch 58/200
137/137 [==============================] - 1s 9ms/step - loss: 110239680.0000 - val_loss:
11555053.0000
Epoch 59/200
137/137 [==============================] - 1s 9ms/step - loss: 60662432.0000 - val_loss:
36634192.0000
Epoch 60/200
137/137 [==============================] - 1s 8ms/step - loss: 334347648.0000 - val_loss:
1405884416.0000
Epoch 61/200
137/137 [==============================] - 1s 9ms/step - loss: 185548288.0000 - val_loss:
131896416.0000
Epoch 62/200
137/137 [==============================] - 1s 10ms/step - loss: 78327400.0000 - val_loss:
13371609.0000
Epoch 63/200
137/137 [==============================] - 1s 10ms/step - loss: 62639952.0000 - val_loss:
11524613.0000
Epoch 64/200
137/137 [==============================] - 1s 9ms/step - loss: 61932388.0000 - val_loss:
14270898.0000
Epoch 65/200
137/137 [==============================] - 1s 9ms/step - loss: 68878512.0000 - val_loss:
9287247.0000
Epoch 66/200
137/137 [==============================] - 1s 9ms/step - loss: 102471848.0000 - val_loss:
14637601.0000
Epoch 67/200
137/137 [==============================] - 1s 9ms/step - loss: 57736904.0000 - val_loss:
12210730.0000
Epoch 68/200
137/137 [==============================] - 1s 10ms/step - loss: 63075076.0000 - val_loss:
6945786.0000
Epoch 69/200
137/137 [==============================] - 1s 9ms/step - loss: 93494448.0000 - val_loss:
27888694.0000
Epoch 70/200
137/137 [==============================] - 1s 9ms/step - loss: 83375064.0000 - val_loss:
12793161.0000
Epoch 71/200
137/137 [==============================] - 1s 9ms/step - loss: 84538856.0000 - val_loss:
8234606.5000
Epoch 72/200
137/137 [==============================] - 1s 9ms/step - loss: 77049176.0000 - val_loss:
17141082.0000
Epoch 73/200
137/137 [==============================] - 1s 8ms/step - loss: 135784816.0000 - val_loss:
38994200.0000
Epoch 74/200
137/137 [==============================] - 1s 9ms/step - loss: 77854112.0000 - val_loss:
205199680.0000
Epoch 75/200
137/137 [==============================] - 1s 9ms/step - loss: 141531072.0000 - val_loss:
28553942.0000
Epoch 76/200
137/137 [==============================] - 1s 9ms/step - loss: 90667976.0000 - val_loss:
63159756.0000
Epoch 77/200
137/137 [==============================] - 1s 9ms/step - loss: 172776384.0000 - val_loss:
37066856.0000
Epoch 78/200
137/137 [==============================] - 1s 8ms/step - loss: 161817264.0000 - val_loss:
13693541.0000
Epoch 79/200
137/137 [==============================] - 1s 9ms/step - loss: 58715684.0000 - val_loss:
10814740.0000
Epoch 80/200
137/137 [==============================] - 1s 9ms/step - loss: 81302952.0000 - val_loss:
83120744.0000
Epoch 81/200
137/137 [==============================] - 1s 9ms/step - loss: 61333520.0000 - val_loss:
```

```
5903852.0000
Epoch 82/200
137/137 [==============================] - 1s 9ms/step - loss: 80825840.0000 - val_loss:
175288448.0000
Epoch 83/200
137/137 [==============================] - 1s 9ms/step - loss: 85480288.0000 - val_loss:
8268953.0000
Epoch 84/200
137/137 [==============================] - 1s 9ms/step - loss: 67892296.0000 - val_loss:
10454562.0000
Epoch 85/200
137/137 [==============================] - 1s 10ms/step - loss: 84570168.0000 - val_loss:
77367064.0000
Epoch 86/200
137/137 [==============================] - 1s 9ms/step - loss: 138476496.0000 - val_loss:
514410752.0000
Epoch 87/200
137/137 [==============================] - 1s 9ms/step - loss: 133651320.0000 - val_loss:
7952011.5000
Epoch 88/200
137/137 [==============================] - 1s 9ms/step - loss: 61185460.0000 - val_loss:
7273567.5000
Epoch 89/200
137/137 [==============================] - 1s 9ms/step - loss: 59586096.0000 - val_loss:
100760768.0000
Epoch 90/200
137/137 [==============================] - 1s 9ms/step - loss: 70732312.0000 - val_loss:
8097661.0000
Epoch 91/200
137/137 [==============================] - 1s 9ms/step - loss: 81595704.0000 - val_loss:
10145113.0000
Epoch 92/200
137/137 [==============================] - 1s 9ms/step - loss: 65970036.0000 - val_loss:
17585730.0000
Epoch 93/200
137/137 [==============================] - 1s 9ms/step - loss: 61345684.0000 - val_loss:
12428767.0000
Epoch 94/200
137/137 [==============================] - 1s 9ms/step - loss: 66328268.0000 - val_loss:
95092984.0000
Epoch 95/200
137/137 [==============================] - 1s 9ms/step - loss: 102477256.0000 - val_loss:
10510416.0000
Epoch 96/200
137/137 [==============================] - 1s 9ms/step - loss: 178444544.0000 - val_loss:
55917496.0000
Epoch 97/200
137/137 [==============================] - 1s 9ms/step - loss: 73158528.0000 - val_loss:
9859935.0000
Epoch 98/200
137/137 [==============================] - 1s 8ms/step - loss: 100933288.0000 - val_loss:
242397712.0000
Epoch 99/200
137/137 [==============================] - 1s 9ms/step - loss: 178701104.0000 - val_loss:
46047640.0000
Epoch 100/200
137/137 [==============================] - 1s 9ms/step - loss: 148290912.0000 - val_loss:
10336379.0000
Epoch 101/200
137/137 [==============================] - 1s 10ms/step - loss: 58924236.0000 - val_loss:
30891278.0000
Epoch 102/200
137/137 [==============================] - 1s 9ms/step - loss: 60148340.0000 - val_loss:
8003301.0000
Epoch 103/200
137/137 [==============================] - 1s 10ms/step - loss: 168892720.0000 - val_loss
: 405060928.0000
Epoch 104/200
137/137 [==============================] - 1s 9ms/step - loss: 138168864.0000 - val_loss:
15031459.0000
Epoch 105/200
137/137 [==============================] - 1s 9ms/step - loss: 56642436.0000 - val_loss:
```

```
7524575.0000
Epoch 106/200
137/137 [==============================] - 1s 9ms/step - loss: 56813740.0000 - val_loss:
15947397.0000
Epoch 107/200
137/137 [==============================] - 1s 9ms/step - loss: 59274284.0000 - val_loss:
8282829.5000
Epoch 108/200
137/137 [==============================] - 1s 9ms/step - loss: 63782556.0000 - val_loss:
34394472.0000
Epoch 109/200
137/137 [==============================] - 1s 9ms/step - loss: 152552512.0000 - val_loss:
10529767.0000
Epoch 110/200
137/137 [==============================] - 1s 9ms/step - loss: 60387920.0000 - val_loss:
8621961.0000
Epoch 111/200
137/137 [==============================] - 1s 10ms/step - loss: 83724424.0000 - val_loss:
13038866.0000
Epoch 112/200
137/137 [==============================] - 1s 9ms/step - loss: 58756808.0000 - val_loss:
23403062.0000
Epoch 113/200
137/137 [==============================] - 1s 10ms/step - loss: 69079672.0000 - val_loss:
57447392.0000
Epoch 114/200
137/137 [==============================] - 1s 8ms/step - loss: 60867388.0000 - val_loss:
19497202.0000
Epoch 115/200
137/137 [==============================] - 1s 9ms/step - loss: 109853544.0000 - val_loss:
9890296.0000
Epoch 116/200
137/137 [==============================] - 1s 9ms/step - loss: 85403152.0000 - val_loss:
74534456.0000
Epoch 117/200
137/137 [==============================] - 1s 10ms/step - loss: 82253552.0000 - val_loss:
9558048.0000
Epoch 118/200
137/137 [==============================] - 1s 9ms/step - loss: 63087440.0000 - val_loss:
98804888.0000
Epoch 119/200
137/137 [==============================] - 1s 9ms/step - loss: 68213632.0000 - val_loss:
21882546.0000
Epoch 120/200
137/137 [==============================] - 1s 9ms/step - loss: 65192740.0000 - val_loss:
8138816.5000
Epoch 121/200
137/137 [==============================] - 1s 9ms/step - loss: 62542104.0000 - val_loss:
7025317.0000
Epoch 122/200
137/137 [==============================] - 1s 9ms/step - loss: 71526904.0000 - val_loss:
8287000.0000
Epoch 123/200
137/137 [==============================] - 1s 9ms/step - loss: 108172832.0000 - val_loss:
393121440.0000
Epoch 124/200
137/137 [==============================] - 1s 9ms/step - loss: 137398416.0000 - val_loss:
11394859.0000
Epoch 125/200
137/137 [==============================] - 1s 9ms/step - loss: 117881824.0000 - val_loss:
22863496.0000
Epoch 126/200
137/137 [==============================] - 1s 9ms/step - loss: 67889512.0000 - val_loss:
26526922.0000
Epoch 127/200
137/137 [==============================] - 1s 10ms/step - loss: 364523040.0000 - val_loss
: 1208386816.0000
Epoch 128/200
137/137 [==============================] - 1s 9ms/step - loss: 114847608.0000 - val_loss:
9454564.0000
Epoch 129/200
137/137 [==============================] - 1s 9ms/step - loss: 62832816.0000 - val_loss:
```

```
13307615.0000
Epoch 130/200
137/137 [==============================] - 1s 9ms/step - loss: 66262956.0000 - val_loss:
15081109.0000
Epoch 131/200
137/137 [==============================] - 1s 10ms/step - loss: 58613232.0000 - val_loss:
20763104.0000
Epoch 132/200
137/137 [==============================] - 1s 10ms/step - loss: 67624448.0000 - val_loss:
13947488.0000
Epoch 133/200
137/137 [==============================] - 1s 9ms/step - loss: 56828368.0000 - val_loss:
11276601.0000
Epoch 134/200
137/137 [==============================] - 1s 9ms/step - loss: 64485484.0000 - val_loss:
13741819.0000
Epoch 135/200
137/137 [==============================] - 1s 9ms/step - loss: 66298284.0000 - val_loss:
323828480.0000
Epoch 136/200
137/137 [==============================] - 1s 9ms/step - loss: 102513096.0000 - val_loss:
6385809.5000
Epoch 137/200
137/137 [==============================] - 1s 10ms/step - loss: 62480328.0000 - val_loss:
9202352.0000
Epoch 138/200
137/137 [==============================] - 1s 9ms/step - loss: 66875212.0000 - val_loss:
12720641.0000
Epoch 139/200
137/137 [==============================] - 1s 10ms/step - loss: 73018320.0000 - val_loss:
6709848.0000
Epoch 140/200
137/137 [==============================] - 1s 9ms/step - loss: 130419248.0000 - val_loss:
130127344.0000
Epoch 141/200
137/137 [==============================] - 1s 9ms/step - loss: 71566792.0000 - val_loss:
7679151.5000
Epoch 142/200
137/137 [==============================] - 1s 9ms/step - loss: 59647604.0000 - val_loss:
19021610.0000
Epoch 143/200
137/137 [==============================] - 1s 10ms/step - loss: 191915536.0000 - val_loss
: 63771420.0000
Epoch 144/200
137/137 [==============================] - 1s 9ms/step - loss: 79510736.0000 - val_loss:
7909455.5000
Epoch 145/200
137/137 [==============================] - 1s 9ms/step - loss: 65774844.0000 - val_loss:
24658410.0000
Epoch 146/200
137/137 [==============================] - 1s 9ms/step - loss: 86618680.0000 - val_loss:
84499560.0000
Epoch 147/200
137/137 [==============================] - 1s 9ms/step - loss: 74378744.0000 - val_loss:
13082152.0000
Epoch 148/200
137/137 [==============================] - 1s 9ms/step - loss: 61654536.0000 - val_loss:
14000613.0000
Epoch 149/200
137/137 [==============================] - 1s 9ms/step - loss: 84202688.0000 - val_loss:
206065328.0000
Epoch 150/200
137/137 [==============================] - 1s 10ms/step - loss: 94653968.0000 - val_loss:
1017412352.0000
Epoch 151/200
137/137 [==============================] - 1s 10ms/step - loss: 420120960.0000 - val_loss
: 14251124.0000
Epoch 152/200
137/137 [==============================] - 1s 9ms/step - loss: 57901216.0000 - val_loss:
20074440.0000
Epoch 153/200
137/137 [==============================] - 1s 9ms/step - loss: 111227104.0000 - val_loss:
```

```
47782296.0000
Epoch 154/200
137/137 [==============================] - 1s 9ms/step - loss: 142054960.0000 - val_loss:
95668000.0000
Epoch 155/200
137/137 [==============================] - 1s 9ms/step - loss: 113136472.0000 - val_loss:
11621177.0000
Epoch 156/200
137/137 [==============================] - 1s 9ms/step - loss: 67377432.0000 - val_loss:
8749787.0000
Epoch 157/200
137/137 [==============================] - 1s 9ms/step - loss: 61475732.0000 - val_loss:
10758383.0000
Epoch 158/200
137/137 [==============================] - 1s 9ms/step - loss: 66195052.0000 - val_loss:
14907318.0000
Epoch 159/200
137/137 [==============================] - 1s 9ms/step - loss: 74837856.0000 - val_loss:
7764750.5000
Epoch 160/200
137/137 [==============================] - 1s 10ms/step - loss: 68924360.0000 - val_loss:
9464890.0000
Epoch 161/200
137/137 [==============================] - 1s 9ms/step - loss: 64032932.0000 - val_loss:
7779522.5000
Epoch 162/200
137/137 [==============================] - 1s 9ms/step - loss: 71777336.0000 - val_loss:
53113992.0000
Epoch 163/200
137/137 [==============================] - 1s 9ms/step - loss: 141167040.0000 - val_loss:
64719176.0000
Epoch 164/200
137/137 [==============================] - 1s 9ms/step - loss: 59592940.0000 - val_loss:
19377646.0000
Epoch 165/200
137/137 [==============================] - 1s 9ms/step - loss: 150134080.0000 - val_loss:
37085404.0000
Epoch 166/200
137/137 [==============================] - 1s 9ms/step - loss: 83781752.0000 - val_loss:
11518783.0000
Epoch 167/200
137/137 [==============================] - 1s 9ms/step - loss: 61296384.0000 - val_loss:
20235700.0000
Epoch 168/200
137/137 [==============================] - 1s 10ms/step - loss: 68638720.0000 - val_loss:
16223900.0000
Epoch 169/200
137/137 [==============================] - 1s 9ms/step - loss: 62109776.0000 - val_loss:
52492488.0000
Epoch 170/200
137/137 [==============================] - 1s 9ms/step - loss: 146989024.0000 - val_loss:
8098269.5000
Epoch 171/200
137/137 [==============================] - 1s 10ms/step - loss: 54864956.0000 - val_loss:
6880201.0000
Epoch 172/200
137/137 [==============================] - 1s 10ms/step - loss: 55007568.0000 - val_loss:
6747706.0000
Epoch 173/200
137/137 [==============================] - 1s 10ms/step - loss: 60053216.0000 - val_loss:
24256138.0000
Epoch 174/200
137/137 [==============================] - 1s 10ms/step - loss: 64411388.0000 - val_loss:
12982551.0000
Epoch 175/200
137/137 [==============================] - 1s 10ms/step - loss: 93684072.0000 - val_loss:
251296432.0000
Epoch 176/200
137/137 [==============================] - 1s 9ms/step - loss: 86389800.0000 - val_loss:
9613698.0000
Epoch 177/200
137/137 [==============================] - 1s 9ms/step - loss: 77716592.0000 - val_loss:
```

```
9163225.0000
Epoch 178/200
137/137 [==============================] - 1s 9ms/step - loss: 62797164.0000 - val_loss:
14359260.0000
Epoch 179/200
137/137 [==============================] - 1s 9ms/step - loss: 66804536.0000 - val_loss:
16269761.0000
Epoch 180/200
137/137 [==============================] - 1s 10ms/step - loss: 70322408.0000 - val_loss:
10259558.0000
Epoch 181/200
137/137 [==============================] - 1s 9ms/step - loss: 67828576.0000 - val_loss:
10743047.0000
Epoch 182/200
137/137 [==============================] - 1s 10ms/step - loss: 90949560.0000 - val_loss:
216251856.0000
Epoch 183/200
137/137 [==============================] - 1s 9ms/step - loss: 80443864.0000 - val_loss:
8591476.0000
Epoch 184/200
137/137 [==============================] - 1s 10ms/step - loss: 61006088.0000 - val_loss:
6104456.0000
Epoch 185/200
137/137 [==============================] - 1s 10ms/step - loss: 207037568.0000 - val_loss
: 147842624.0000
Epoch 186/200
137/137 [==============================] - 1s 9ms/step - loss: 123477352.0000 - val_loss:
11336923.0000
Epoch 187/200
137/137 [==============================] - 1s 9ms/step - loss: 96143904.0000 - val_loss:
37223264.0000
Epoch 188/200
137/137 [==============================] - 1s 9ms/step - loss: 63264888.0000 - val_loss:
11898750.0000
Epoch 189/200
137/137 [==============================] - 1s 10ms/step - loss: 77065352.0000 - val_loss:
29991924.0000
Epoch 190/200
137/137 [==============================] - 1s 10ms/step - loss: 76601680.0000 - val_loss:
50097868.0000
Epoch 191/200
137/137 [==============================] - 1s 11ms/step - loss: 67671040.0000 - val_loss:
16082324.0000
Epoch 192/200
137/137 [==============================] - 1s 10ms/step - loss: 60178236.0000 - val_loss:
14219319.0000
Epoch 193/200
137/137 [==============================] - 1s 9ms/step - loss: 61541448.0000 - val_loss:
6480240.0000
Epoch 194/200
137/137 [==============================] - 1s 10ms/step - loss: 55418712.0000 - val_loss:
7851633.5000
Epoch 195/200
137/137 [==============================] - 1s 10ms/step - loss: 63877428.0000 - val_loss:
8781068.0000
Epoch 196/200
137/137 [==============================] - 1s 10ms/step - loss: 103701192.0000 - val_loss
: 34784324.0000
Epoch 197/200
137/137 [==============================] - 1s 10ms/step - loss: 71992680.0000 - val_loss:
7168349.5000
Epoch 198/200
137/137 [==============================] - 1s 10ms/step - loss: 72885096.0000 - val_loss:
64543024.0000
Epoch 199/200
137/137 [==============================] - 1s 9ms/step - loss: 67068552.0000 - val_loss:
8848151.0000
Epoch 200/200
137/137 [==============================] - 1s 9ms/step - loss: 54754004.0000 - val_loss:
10827558.0000
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f5f32ea8dd0>
```

In [ ]:

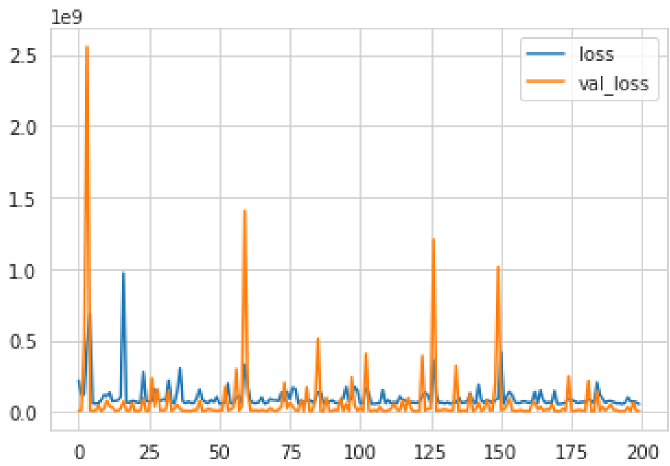
```
loss = pd.DataFrame(model.history.history)
```

In [ ]:

```
loss.plot()
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f32f46fd0>
```



In [ ]:

```
test_ann_pred = model.predict(X_test)
```

In [ ]:

```
print('Test data evaluation:\n----------------------------')
print_metrics(y_test, test_ann_pred)
```

```
Test data evaluation:
----------------------------
MAE 901.4310343650818
MSE 10827562.999530697
RMSE 3290.5262496340456
R2_squared 0.9989256563011628
--------------------------------------
```

Artificial neural networks also given best fit with R2_score 0.999, which seems a better model.

Getting the Data Frame of all the results.

In [ ]:

```
def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```

In [ ]:

```
df1 = pd.DataFrame(data=[["Linear Regression", *evaluate(y_test, test_pred)]], columns=[
'Model','MAE','MSE','RMSE','R2_SCORE'])
```

In [ ]:

```
df2 = pd.DataFrame(data=[["Decision Tree", *evaluate(y_test, test_dtm_pred)]], columns=[
'Model','MAE','MSE','RMSE','R2_SCORE'])
```

```
df1 = df1.append(df2, ignore_index=True)
```

In [ ]:

```
df3 = pd.DataFrame(data=[["Random Forest", *evaluate(y_test, test_rf_pred)]], columns=['
Model','MAE','MSE','RMSE','R2_SCORE'])
df1 = df1.append(df3, ignore_index=True)
```

In [ ]:

```
df4 = pd.DataFrame(data=[["Artificial Neural Networks", *evaluate(y_test, test_ann_pred)
]], columns=['Model','MAE','MSE','RMSE','R2_SCORE'])
df1 = df1.append(df4, ignore_index=True)
```

In [ ]:

```
df1.head()
```

Out[ ]:

	Model	MAE	MSE	RMSE	R2_SCORE
0	**Linear Regression**	**311.184926**	**5.359971e+06**	**2315.161127**	**0.999468**
1	**Decision Tree**	**9125.751203**	**4.465480e+08**	**21131.683242**	**0.955692**
2	**Random Forest**	**489.739860**	**1.360519e+07**	**3688.520861**	**0.998650**
3	**Artificial Neural Networks**	**901.431034**	**1.082756e+07**	**3290.526250**	**0.998926**

Out of all the models the Linear Regression model is the best fit for the data and gives the best results for predicting the impact of the tweets with the features concerned.

In [ ]:

```

```

Assignment (Impact of Tweets)

As shown above I completed my assignment and it is written well how I performed all the tasks, from importing the data to predicting the correlated features. I have trained the data on models like Linear Regression, Decision Tree, Random Forests, and Artificial Neural Networks. Out of all the models Linear Regression have performed very well and the data given, is fitted perfectly to the model and predicted the best results.

Report:

The training error rates I obtained are:

Linear Regression:

```
Training data evaluation:
----------------------------
MAE 390.49829714982894
MSE 39866179.92028137
RMSE 6313.967050934093
R2_squared 0.9963284216068522
```

Decision Tree:

```
Training data evaluation:
----------------------------
MAE 9297.597976476207
MSE 543755944.42697
RMSE 23318.575094267017
R2_squared 0.949921397517997
```

Random forests:

```
Training data evaluation:
----------------------------
MAE 214.1747494558196
MSE 4507666.964254432
RMSE 2123.126695290329
R2_squared 0.9995848548152204
```

Testing error rates:

Linear Regression:

```
Test data evaluation:
----------------------------
```

```
MAE 311.18492577240653
MSE 5359971.045152342
RMSE 2315.1611272549353
R2_squared 0.9994681673873835
```

Decision Tree:

```
Test data evaluation:
----------------------------
MAE 9125.75120292198
MSE 446548036.65582645
RMSE 21131.68324236918
R2_squared 0.9556921470297507
```

Random Forests:

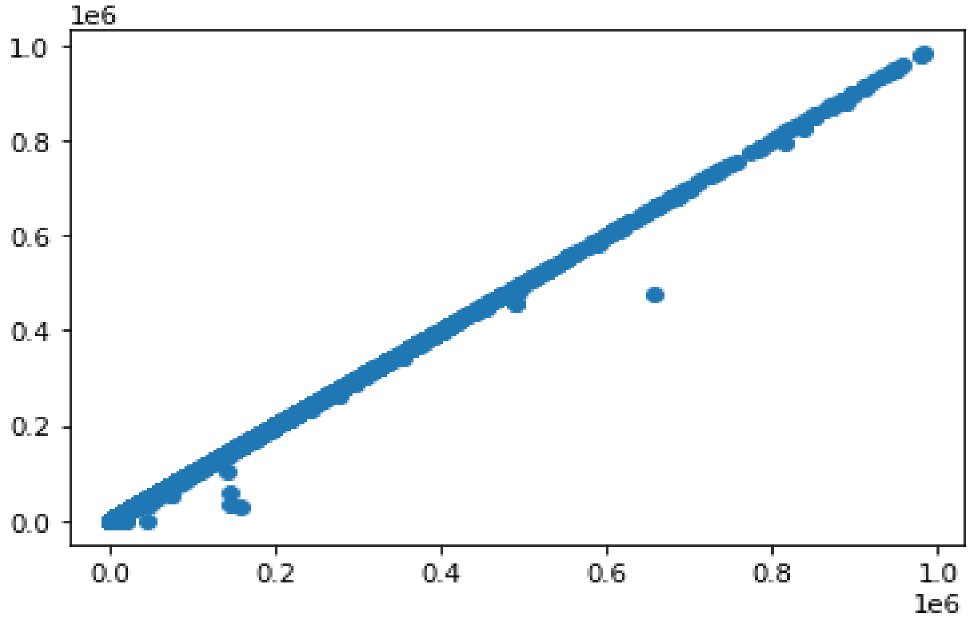
```
Test data evaluation:
----------------------------
MAE 489.73985983942964
MSE 13605186.139588151
RMSE 3688.520860668698
R2_squared 0.99865005209379
```

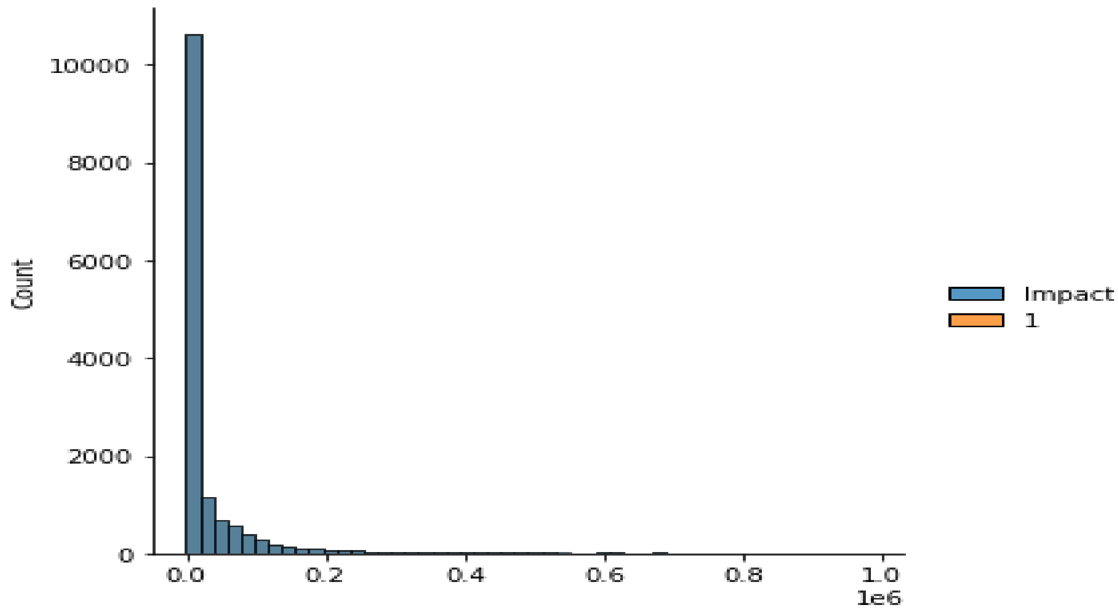
Artificial Neural networks:

```
Test data evaluation:
----------------------------
MAE 901.4310343650818
MSE 10827562.999530697
RMSE 3290.5262496340456
R2_squared 0.9989256563011628
```

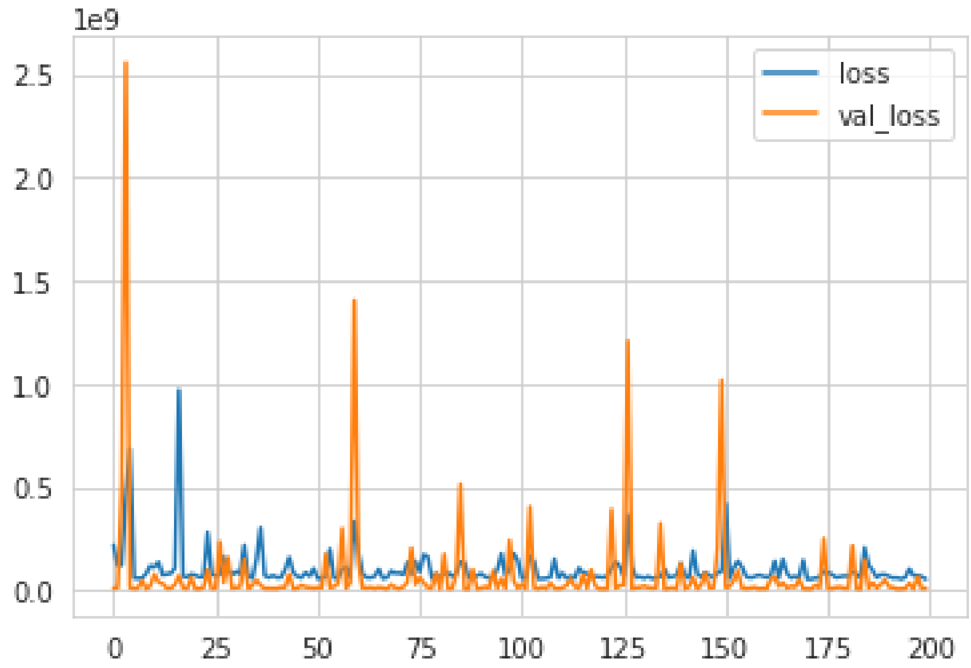
Graphs that show performance on both training and testing data.

Linear Regression:





Artificial Neural networks:



- The reason I got results is the data (features) given to the model are correlated to each other and the impact variable in highly dependent on these features, so the model trained well and given the best results. Given the accuracy almost high, I.e., R2_score is nearly to one (0.9999) and the

Root mean square error (rmse is very low). These results shows it's the best.

- The model is pretty much fast as it is predicting the results in less than 5 seconds.
- I have chosen the learning rates to give the best loss without sacrificing the training time, I have started with, higher rates of 0.1 and tried for 0.01, 0.001 etc., and finally I have taken 0.0005 which is giving the best loss for my model.
- The best dropout rates for any model is 0.1 so, I take it for the best results.
- The Linear Regression algorithm performed best among all the models.
- I have defined the best model by predicting the R2_score. If R2_score is 1, it indicates the best, other than this the model has high rate of errors. So linear regression got best R2_score = 0.9999 which is nearly 1.