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

data_df=pd.read_excel("/content/marks_vs_rank.xlsx")

data_df.head(32)
```

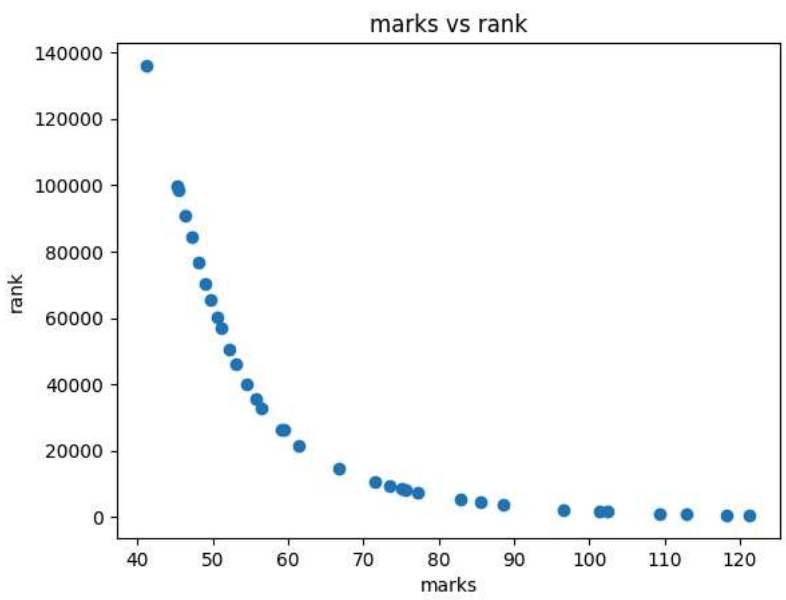
0	41.253	136070
1	45.283	99667
2	45.455	98560
3	46.322	90856
4	47.170	84353
5	48.983	70147
6	48.120	76847
7	49.665	65399
8	50.520	60078
9	51.064	56824
10	52.189	50704
11	53.138	46061
12	54.421	40228
13	55.674	35520
14	56.477	32818
15	59.477	26146
16	59.074	26146
17	61.363	21509
18	66.789	14607
19	71.491	10751
20	73.424	9496
21	75.153	8570
22	75.640	8319
23	77.207	7555
24	82.864	5400
25	85.613	4652
26	88.614	3822
27	96.601	2345
28	101.370	1715
29	102.330	1620
30	109.401	1082
31	112.966	879

Next steps: Generate code with data_df View recommended plots

```
data_df.corr()
```

	marks	rank
marks	1.000000	-0.802375
rank	-0.802375	1.000000

```
x=data_df['marks'].values.reshape(-1,1)
plt.xlabel("marks")
plt.ylabel("rank")
plt.title("marks vs rank")
y=data_df['rank'].values.reshape(-1,1)
# plt.plot(x,y)
plt.scatter(x,y)
plt.show()
```



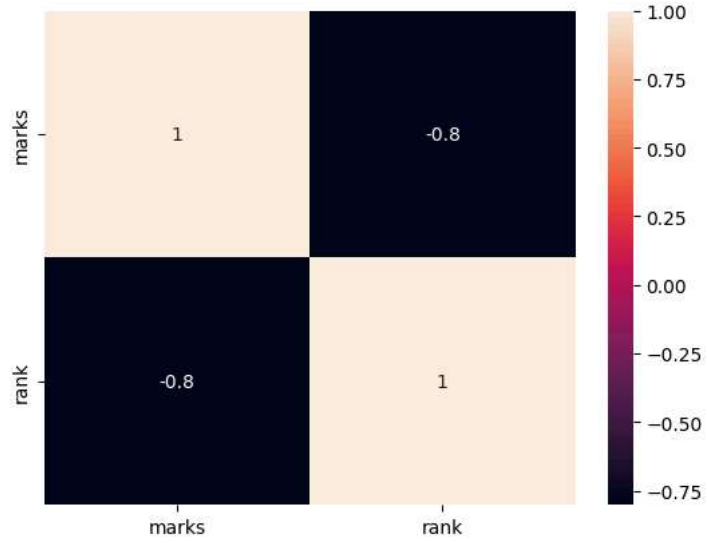
TS EAMCET MARKS VS RANK

THE USER INPUT MARKS WE PREDICT RANK

as we can see there is co realation but the graph is not leniar the lago such as leniar reggressiin svm wont work nad as the distance bewteen the data points is so high we can also do prediction by using accuracy mesures we shold draw graphs manually between test data and the predicted data

```
sns.heatmap(data_df.corr(),annot=True)
```

```
<Axes: >
```




```
data_df.describe()
```

	marks	rank
count	34.00000	34.000000
mean	70.13600	35293.588235
std	23.81238	36639.117219
min	41.25300	562.000000
25%	50.65600	4839.000000
50%	60.42000	23827.500000
75%	84.92575	59264.500000
max	121.29600	136070.000000

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34 entries, 0 to 33
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   marks   34 non-null     float64
 1   rank    34 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 672.0 bytes
```

```
data_df['rank']=data_df['rank'].astype(float)
```

```
y=data_df['rank'].values.reshape(-1,1)
x=data_df['marks'].values.reshape(-1,1)
```

```
print(x)
x.shape
```

```
[[ 41.253]
 [ 45.283]
 [ 45.455]
 [ 46.322]
 [ 47.17 ]
 [ 48.983]
 [ 48.12 ]
 [ 49.665]
 [ 50.52 ]
 [ 51.064]
 [ 52.189]
 [ 53.138]
 [ 54.421]
```

```
       [ 55.674]
       [ 56.477]
       [ 59.477]
       [ 59.074]
       [ 61.363]
       [ 66.789]
       [ 71.491]
       [ 73.424]
       [ 75.153]
       [ 75.64 ]
       [ 77.207]
       [ 82.864]
       [ 85.613]
       [ 88.614]
       [ 96.601]
       [101.37 ]
       [102.33 ]
       [109.401]
       [112.966]
       [118.217]
       [121.296]]
      (34, 1)
```

```
print(y)
y.shape
```

```
[[136070.]
 [ 99667.]
 [ 98560.]
 [ 90856.]
 [ 84353.]
 [ 70147.]
 [ 76847.]
 [ 65399.]
 [ 60078.]
 [ 56824.]
 [ 50704.]
 [ 46061.]
 [ 40228.]
 [ 35520.]
 [ 32818.]
 [ 26146.]
 [ 26146.]
 [ 21509.]
 [ 14607.]
 [ 10751.]
 [  9496.]
 [  8570.]
 [  8319.]
 [  7555.]
 [  5400.]
 [  4652.]
 [  3822.]
 [  2345.]
 [  1715.]
 [  1620.]
 [  1082.]
 [   879.]
 [   674.]
 [   562.]]
(34, 1)
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(27, 1)
(27, 1)
(7, 1)
(7, 1)
```

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

```
reg=LinearRegression()

model=reg.fit(x_train,y_train)

from sklearn.metrics import accuracy_score

y_pred=model.predict(x_test)

y_pred.shape
```

```
(7, 1)
```

```
x_test.shape
```

```
(7, 1)
```

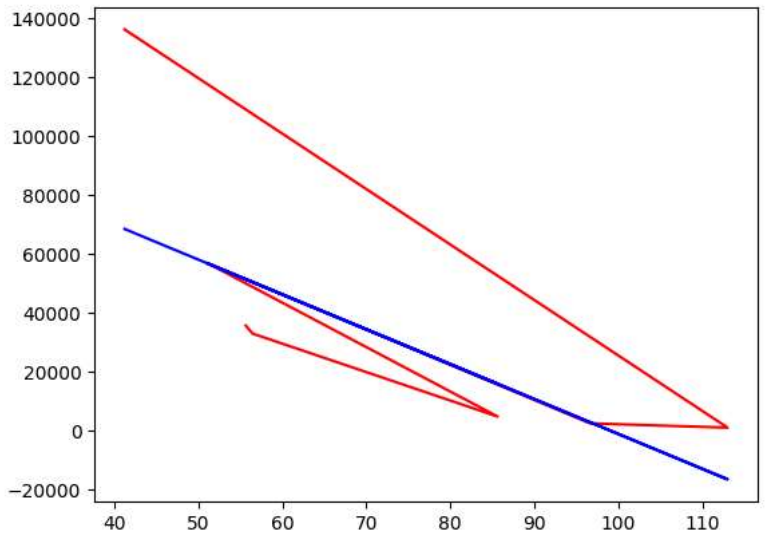
```
from sklearn.metrics import mean_squared_error, r2_score

# Assuming y_test is your actual target values and y_pred is your model's predictions
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)  # Calculate RMSE
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 796041370.8903185
Root Mean Squared Error: 28214.20512597012
R-squared: 0.5951850219278476
```

```
# drawing graph between values
plt.plot(x_test,y_test,color='red')
plt.plot(x_test,y_pred,color='blue')
plt.show()
```




```
# using suppoer vector
from sklearn.svm import SVR
model1=SVR()
model1.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversio
  y = column_or_1d(y, warn=True)
```

```
▾ SVR
SVR()
```

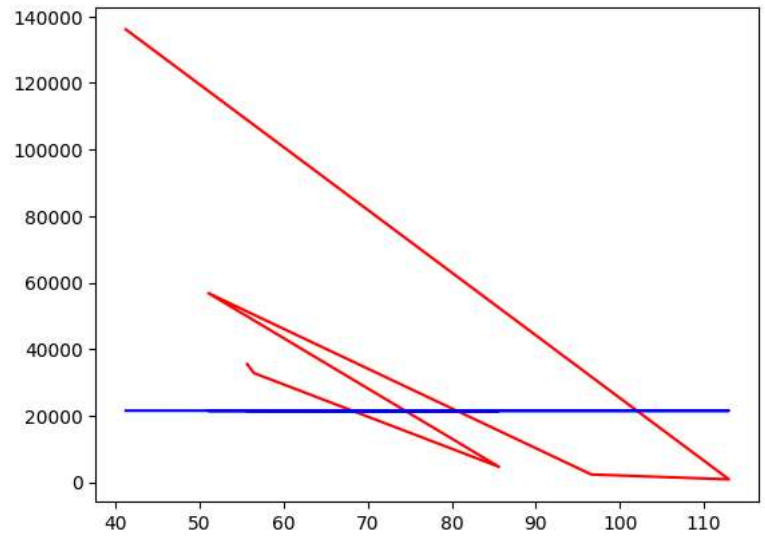
```
y_svr_pred=model1.predict(x_test)


mse = mean_squared_error(y_test, y_svr_pred)
rmse = np.sqrt(mse)  # Calculate RMSE
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 2252816663.855718
Root Mean Squared Error: 47463.845860356894
R-squared: 0.5951850219278476
```

```
plt.plot(x_test,y_test,color='red')
plt.plot(x_test,y_svr_pred,color='blue')
plt.show()
```




```
# using random forest
from sklearn.ensemble import RandomForestRegressor
model2=RandomForestRegressor()
model2.fit(x_train,y_train)
```

```
<ipython-input-29-4366add4e094>:4: DataConversionWarning: A column-vector y was passed
  model2.fit(x_train,y_train)
▾ RandomForestRegressor
RandomForestRegressor()
```

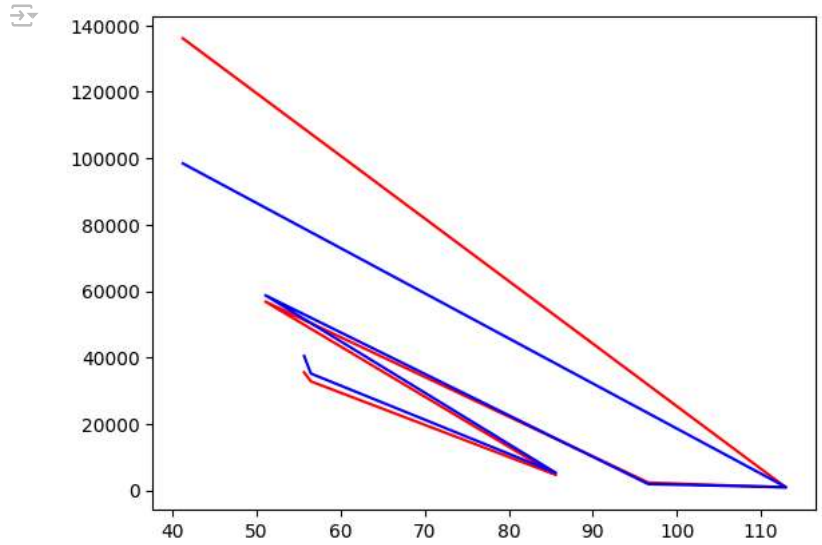
```
Y_random_pred=model2.predict(x_test)


mse = mean_squared_error(y_test, Y_random_pred)
rmse = np.sqrt(mse)  # Calculate RMSE
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 207405569.0905714
Root Mean Squared Error: 14401.58217317012
R-squared: 0.5951850219278476
```

```
# dawing graph
plt.plot(x_test,y_test,color='red')
plt.plot(x_test,Y_random_pred,color='blue')
plt.show()
```




```
print(Y_random_pred)
```

```
[40439.25 35168.49  5275.98 58729.01  1894.08   983.69 98415.22]
```

```
print(y_test)
```

```
[[ 35520.]
 [ 32818.]
 [  4652.]
 [ 56824.]
 [  2345.]
 [   879.]
 [136070.]]
```

```
Y_random_pred=Y_random_pred.reshape(-1,1)
```

```
print(Y_random_pred)
```

```
[[40439.25]
 [35168.49]
 [ 5275.98]
 [58729.01]
 [ 1894.08]
 [  983.69]
 [98415.22]]
```

Double-click (or enter) to edit

```
from sklearn import tree
```

```
model3 = tree.DecisionTreeClassifier()
```

```
model3.fit(x_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

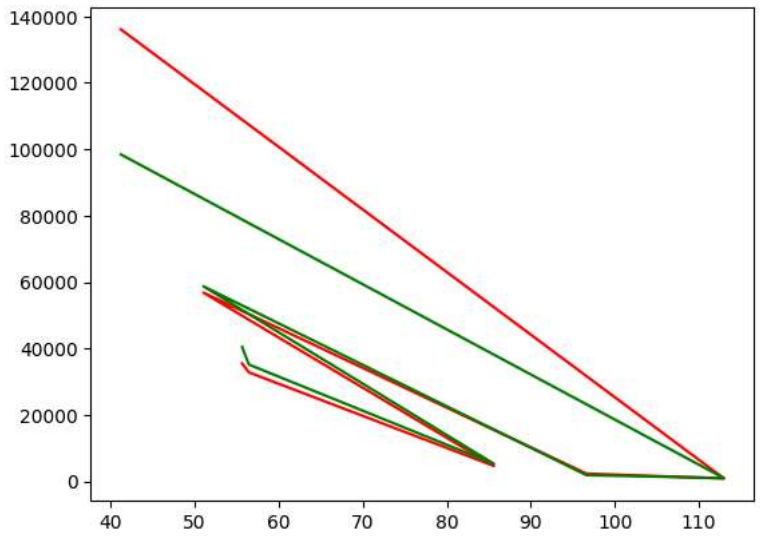
```
Y_desion_tree=model3.predict(x_test)
```

```
mse = mean_squared_error(y_test, Y_desion_tree)
rmse = np.sqrt(mse)  # Calculate RMSE
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 201976843.14285713
Root Mean Squared Error: 14211.85572481149
R-squared: 0.5951850219278476
```

```
# dawing graph
plt.plot(x_test,y_test,color='red')
plt.plot(x_test,Y_random_pred,color='green')
# plt.plot(x_test,Y_desion_tree,color='blue')
plt.show()
```




```
print(Y_desion_tree,"/n/n",Y_random_pred,"/n/n",y_test)
```

```
[40228. 40228.  5400. 60078.  1715.  1082. 99667.] /n/n [[40439.25]
 [35168.49]
 [ 5275.98]
 [58729.01]
 [ 1894.08]
 [  983.69]
 [98415.22]] /n/n [[ 35520.]
 [ 32818.]
 [  4652.]
 [ 56824.]
 [  2345.]
 [   879.]
 [136070.]]
```

```
print(Y_random_pred)
print(x_test)
print(y_test)
```

```
[[40439.25]
 [35168.49]
 [ 5275.98]
 [58729.01]
 [ 1894.08]
 [  983.69]
 [98415.22]]
[[ 55.674]
 [ 56.477]
 [ 85.613]
 [ 51.064]
 [ 96.601]
 [112.966]
 [ 41.253]]
[[ 35520.]
 [ 32818.]
 [  4652.]
 [ 56824.]
 [  2345.]
 [   879.]
 [136070.]]
```

```
model3.predict([[100]])
```

```
array([1715.])
```

```
model3.fit(x,y)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
model3.predict([[100]])
```

```
array([1715.])
```

```
model3.predict([[47]])
```

```
array([84353.])
```

```
model3.predict([[54]])
```

```
array([40228.])
```