new*

```
1  import pandas as pd
2  import numpy as np
3  import pandas_profiling as pp
4  df=pd.read_csv("/storage/emulated/0/
   jhansi/admission1.csv")
5  print(df.head())
6  print(df.tail())
7  print(df.dtypes)
8  print(df.shape)
9  print(df.info)
10 print(df.describe())
11 Profile = pp.ProfileReport(df)
12 Profile.to_file("/storage/emulated/0/
   jhansi/report1.html")
13 df=df.drop("Research",axis=1)
14 print(df.shape)
15 print(df.head())
16 from sklearn.model_selection import
   train_test_split
17 y=df["Chance_of_Admit"]
18 X=df.drop('Chance_of_Admit',axis=1)
19 X_train,X_test,y_train,y_test =
   train_test_split(X,y,test_size = 0.25)
20 from sklearn.neighbors import
   KNeighborsClassifier
21 knn =
   KNeighborsClassifier(n_neighbors=1)
22 knn.fit(X_train,y_train)
23 y_pred = knn.predict(X_test)
24 from sklearn.metrics import
   confusion_matrix,accuracy_sco
25 conf_mat = confusion_matrix(y_
```

Tab | : | ; | ' | # | (

```
16 train_test_split
17 y=df["Chance_of_Admit"]
18 X=df.drop('Chance_of_Admit',axis=1)
19 X_train,X_test,y_train,y_test =
   train_test_split(X,y,test_size = 0.25)
20 from sklearn.neighbors import
   KNeighborsClassifier
21 knn =
   KNeighborsClassifier(n_neighbors=1)
22 knn.fit(X_train,y_train)
23 y_pred = knn.predict(X_test)
24 from sklearn.metrics import
   confusion_matrix,accuracy_score
25 conf_mat = confusion_matrix(y_test,
   y_pred)
26 print(conf_mat)
27 acc_score = accuracy_score(y_test,
   y_pred)
28 print(acc_score)
29 new_parameters=[[340,120,4.5,5.0,5.0,9.8,
   1]]
30 Chance_of_Admit_predicted=knn.
   predict(new_parameters)
31 print(Chance_of_Admit_predicted)
```

```
#Linear Regression
import pandas as pd
import numpy as np
import pandas_profiling as pp
df=pd.read_csv("/storage/emulated/0/
pinky/admission1.csv")
print(df.head())
print(df.tail())
print(df.dtypes)
print(df.shape)
print(df.info)
print(df.describe())
Profile = pp.ProfileReport(df)
Profile.to_file("/storage/emulated/0/pinky/
report1.html")
df=df.drop("Research",axis=1)
print(df.shape)
print(df.head())
mean_G=df.GRE_Score.mean()
df.GRE_Score=df.GRE_Score.
replace({0:mean_G})
mean_T=df.TOEFL_Score.mean()
df.TOEFL_Score=df.TOEFL_Score.
replace({0:mean_T})
mean_U=df.University_Rating.mean()
df.University_Rating=df.University_Rating.
replace({0:mean_T})
mean_S=df.SOP.mean()
df.SOP=df.SOP.replace({0:mean_S})
mean_L=df.LOR.mean()
df.LOR=df.LOR.replace({0:mean
mean_C=df.CGPA.mean()
```

Tab | : | ; | ' | # | (

new*

```
mean_C=df.CGPA.mean()
df.CGPA=df.CGPA.replace({0:mean_C})
mean_A=df.Chance_of_Admit.mean()
df.Chance_of_Admit=df.Chance_of_Admit.
replace({0:mean_A})
y=df[['Chance_of_Admit']]
X=df.drop('Chance_of_Admit',axis=1)
from sklearn.model_selection import
train_test_split
X_train,X_test,y_train,
y_test=train_test_split(X,y,test_size=0.25,
random_state=1)
#model building
from sklearn.linear_model import
LinearRegression
regression_model=LinearRegression()
print(regression_model.fit(X_train,y_train))
intercept=regression_model.intercept_[0]
print(intercept)
for idx,col_name in enumerate(X_train.
columns):
    print("The co-efficient for {} is {}".
format(col_name,regression_model.
coef_[0][idx]))
#Evaluation metrics
from sklearn.metrics import
mean_squared_error
y_pred=regression_model.predict(X_test)
print(y_pred)
regression_model_mse=mean_sq    d_er
ror(y_pred,y_test)
print(regression_model_mse)
```

Tab | : | ; | ' | # |

new*

```
columns):
    print("The co-efficient for {} is {}".
format(col_name,regression_model.
coef_[0][idx]))
#Evaluation metrics
from sklearn.metrics import
mean_squared_error
y_pred=regression_model.predict(X_test)
print(y_pred)
regression_model_mse=mean_squared_er
ror(y_pred,y_test)
print(regression_model_mse)
import math
mae=math.sqrt(regression_model_mse)
print(mae)
accuracy=regression_model.score(X_test,
y_test)
print(accuracy)
# pre_deployment test
new_parameters=[[340,120,4.5,5.0,5.0,9.8,
1]]
Chance_of_Admit_predicted=regression_
model.predict(new_parameters)
print(Chance_of_Admit_predicted)
```

Tab | : | ; | ' | # | (

new*

```
1  import pandas as pd
2  import numpy as np
3  import pandas_profiling as pp
4  df=pd.read_csv("/storage/emulated/0/
   jhansi/admission1.csv")
5  print(df.head())
6  print(df.tail())
7  print(df.dtypes)
8  print(df.shape)
9  print(df.info)
10 print(df.describe())
11 Profile = pp.ProfileReport(df)
12 Profile.to_file("/storage/emulated/0/
   jhansi/report1.html")
13 df=df.drop("Research",axis=1)
14 print(df.shape)
15 print(df.head())
16 y=df[['Chance_of_Admit']]
17 X=df.drop('Chance_of_Admit',axis=1)
18 from sklearn.model_selection import
   train_test_split
19 X_train,X_test,y_train,y_test =
   train_test_split(X,y,test_size=0.25)
20 from sklearn.tree import
   DecisionTreeClassifier
21 model = DecisionTreeClassifier()
22 print(model.fit(X_train,y_train))
23 y_pred = model.predict(X_test)
24 print(y_pred)
25 print(y_test)
26 from sklearn.metrics import
   accuracy_score,confusion_matri
```

Tab | : | ; | ' | # | (

new*

```
18 from sklearn.model_selection import
   train_test_split
19 X_train,X_test,y_train,y_test =
   train_test_split(X,y,test_size=0.25)
20 from sklearn.tree import
   DecisionTreeClassifier
21 model = DecisionTreeClassifier()
22 print(model.fit(X_train,y_train))
23 y_pred = model.predict(X_test)
24 print(y_pred)
25 print(y_test)
26 from sklearn.metrics import
   accuracy_score,confusion_matrix
27 conf_mat = confusion_matrix(y_pred,
   y_test)
28 acc_score = accuracy_score(y_pred,
   y_test)
29 print(conf_mat)
30 print(acc_score)
31 new_parameters=[[340,120,4.5,5.0,5.0,9.8,
   1]]
32 Chance_of_Admit_predicted=model.
   predict(new_parameters)
33 print(Chance_of_Admit_predicted)
```

Tab | : | ; | ' | # |