```
# -*- coding: utf-8 -*-
"""Project1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1M1nys_UHy_RlniEyqCHORaret3haAXrN
"""

# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline

df=pd.read_csv("/content/train_hm.csv")
df.head()

"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"



"sex = pd.get_dummies(df['Gender'],drop_first=True)"

"df.drop(['Gender','Loan_ID'],axis=1,inplace=True)"
"df = pd.concat([df,sex],axis=1)"
df.head()

"married = pd.get_dummies(df['Married'],drop_first=True)"
"education = pd.get_dummies(df['Education'],drop_first=True)"
"df.drop(['Married','Education'],axis=1,inplace=True)"
"df = pd.concat([df,married,education],axis=1)"
df.head()

"df=df.rename(columns={'Yes':'Married','Male':'Gender'})"
df.head()

df=df.rename(columns={'Sf':'Self_Employed'})
df.head()

"self_emp = pd.get_dummies(df['Self_Employed'],drop_first=True)"
"df.drop(['Self_Employed'],axis=1,inplace=True)"
"df = pd.concat([df,self_emp],axis=1)"
df.head()

df=df.rename(columns={'Yes':'Self_Employed'})
df.head(50)

"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"

"plt.figure(figsize=(12, 7))"
"bx=sns.boxplot(x='Self_Employed',y='LoanAmount',data=df,palette='winter')"
medians=df.groupby(['Self_Employed'])['LoanAmount'].median()
vertical_offset=df['LoanAmount'].median() * 0.05
```

```
for xtick in bx.get_xticks():
"    bx.text(xtick,medians[xtick]+vertical_offset,medians[xtick],horizontalalignment='center',size='x-small',c
olor='w',weight='semibold')"

def impute_LoanAmt(cols):
    Loan = cols[0]
    selfemp = cols[1]

    if pd.isnull(Loan):

        if selfemp == 1:
            return 150
        else:
            return 125

    else:
        return Loan

"df['LoanAmount'] = df[['LoanAmount','Self_Employed']].apply(impute_LoanAmt,axis=1)"
"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"

"sns.countplot(x='Credit_History',data=df,palette='RdBu_r')"

"df['Credit_History'].fillna(1.0,inplace=True)"
"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"

"sns.countplot(x='Loan_Amount_Term',data=df,palette='RdBu_r')"

"df['Loan_Amount_Term'].fillna(360.0,inplace=True)"
"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"

"sns.countplot(x='Dependents',data=df,palette='RdBu_r')"

"df['Dependents'].fillna(0,inplace=True)"
"sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')"

"loanStatus = pd.get_dummies(df['Loan_Status'],drop_first=True)"
"df.drop(['Loan_Status'],axis=1,inplace=True)"
"df = pd.concat([df,loanStatus],axis=1)"
df.head()

"PropArea = pd.get_dummies(df['Property_Area'],drop_first=True)"
"df.drop(['Property_Area'],axis=1,inplace=True)"
"df = pd.concat([df,PropArea],axis=1)"
df.head()

df=df.rename(columns={'Y':'Loan_Status'})
"df=df.replace(to_replace='3+',value=3)"

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
"train=pd.DataFrame(df.drop('Loan_Status',axis=1))"
scaler.fit(train)
scaled_features = scaler.transform(train)
"df_feat = pd.DataFrame(scaled_features,columns=train.columns)"
```

```
df_feat.head()

from sklearn.model_selection import train_test_split
"X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['Loan_Status'],"
                                                test_size=0.30)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
"knn.fit(X_train,y_train)"

"from sklearn.metrics import classification_report,confusion_matrix"
pred = knn.predict(X_test)
"print(confusion_matrix(y_test,pred))"
"print(classification_report(y_test,pred))"

knn.predict([X_test[0]])
X_test[0]

"features=np.array([1,0,3,1,4000,3000,0,1,360,1.0])"
"features = scaler.fit_transform(features.reshape(-1, 1))"
features=features.flatten()
features

error_rate = []

"for i in range(1,40):"

    knn = KNeighborsClassifier(n_neighbors=i)
"    knn.fit(X_train,y_train)"
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

"plt.figure(figsize=(10,6))"
"plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',"
"         markerfacecolor='red', markersize=10)"
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')

knn = KNeighborsClassifier(n_neighbors=27)
"knn.fit(X_train,y_train)"

pred = knn.predict(X_test)
"print(confusion_matrix(y_test,pred))"
"print(classification_report(y_test,pred))"

from sklearn.metrics import accuracy_score
"print(accuracy_score(y_test,pred)*100)"

df['Dependents'] = pd.to_numeric(df['Dependents'])

df.columns

"x=df[['Married', 'Not Graduate', 'Dependents',"
```

" ’Self_Employed’,’ApplicantIncome’,’CoapplicantIncome’,’Semiurban’,’Urban’,’Loan_Amount_Term’,’Credit_History’]]"
y=df[’LoanAmount’]

"X_train,X_test,Y_train,Y_test=train_test_split(x.values,y.values,test_size=0.3,random_state=101)"

"from sklearn.metrics import max_error,explained_variance_score,mean_absolute_error"
"print(max_error(Y_test,pred))"
"print(explained_variance_score(Y_test,pred))"
"print(mean_absolute_error(Y_test,pred))"

from sklearn.ensemble import RandomForestRegressor
"regr = RandomForestRegressor(max_depth=2, random_state=0)"
"regr.fit(X_train,Y_train)"

pred4=regr.predict(X_test)

"print(explained_variance_score(Y_test,pred4))"
"print(max_error(Y_test,pred4))"

"#’Married’, ’Not Graduate’, ’Dependents’,’Self_Employed’,’ApplicantIncome’,’CoapplicantIncome’,’Semiurban’,’Urban’,’Loan_Amount_Term’,’Credit_History’"

"loan=print(regr.predict([[1,0,2,1,12000,10000,1,0,240,1.0]])*12*89)"

"print(regr.predict([[1,0,2,1,5417,4196,0,1,360,1]])*12*89)"

"print((regr.predict([[1,0,1,1,5620,5700,0,1,360,1.0]]))*12*89)"

"print(regr.predict([[0,1,0,0,6000,0,0,1,360,1]])*12*89)"

regr.predict(X_test)