```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score
import joblib
import time
from scipy.stats import uniform
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive"
```

```
# Loading dataset
credit = pd.read_csv("/content/drive/MyDrive/final_proyect/credit_scores.csv",low_memory=False)
df = credit.copy()
```

## Dropping Columns

```
delete_columns = ["Name", "SSN", "ID", "Customer_ID"]
df.drop(delete_columns, axis=1, inplace=True)
df.head()
```

| | Month | Age | Occupation | Annual_Income | Monthly_Inhand_Salary | Num_Bank_Accounts | Num_Credit_Card | Inte: |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **0** | July | 23.0 | Scientist | 19114.12 | 1824.843333 | 3.0 | 4.0 |
| **1** | February | 28.0 | Teacher | 34847.84 | 3037.986667 | 2.0 | 4.0 |
| **2** | May | 28.0 | Teacher | 34847.84 | 3037.986667 | 2.0 | 4.0 |
| **3** | June | 28.0 | Teacher | 34847.84 | 3037.986667 | 2.0 | 4.0 |
| **4** | August | 28.0 | Teacher | 34847.84 | 3037.986667 | 2.0 | 4.0 |

5 rows × 31 columns

## Selecting Numerical and Categorical Data

```
numerical_format=['float64','int']
numerical_data = df.select_dtypes(include=numerical_format).columns
categorical_data = df.select_dtypes(include='object').columns
```

```
print("Numerical Data")
print(numerical_data)
print("")
print("Categorical Data")
print("")
print(categorical_data)
```

```
Numerical Data
Index(['Age', 'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
       'Num_Credit_Card', 'Interest_Rate', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt', 'Credit_Utilization_Ratio',
       'Credit_History_Age', 'Total_EMI_per_month', 'Amount_invested_monthly',
       'Monthly_Balance', 'Count_Auto Loan', 'Count_Credit-Builder Loan',
       'Count_Personal Loan', 'Count_Home Equity Loan', 'Count_Not Specified',
       'Count_Mortgage Loan', 'Count_Student Loan',
       'Count_Debt Consolidation Loan', 'Count_Payday Loan'],
```

```
       'Count_Debt Consolidation Loan', 'Count_Payday Loan'],
      dtype='object')

Categorical Data

Index(['Month', 'Occupation', 'Credit_Mix', 'Payment_of_Min_Amount',
       'Payment_Behaviour', 'Credit_Score'],
      dtype='object')
```

```
# Checking the min, max and mean values for each Numerical category
for col in numerical_data:
    print(f"{col} : Min {df[col].min()} - Max {df[col].max()} - {df[col].mean()}")
```

```
Age : Min 14.0 - Max 95.0 - 32.963502994011975
Annual_Income : Min 7005.93 - Max 24198062.0 - 180449.1165729597
Monthly_Inhand_Salary : Min 303.6454166666666 - Max 15204.633333333331 - 4011.3346753538767
Num_Bank_Accounts : Min -1.0 - Max 1798.0 - 17.670518846741114
Num_Credit_Card : Min 0.0 - Max 1499.0 - 22.108409927846473
Interest_Rate : Min 1.0 - Max 5797.0 - 77.57309661387383
Delay_from_due_date : Min -5.0 - Max 67.0 - 22.085661247342735
Num_of_Delayed_Payment : Min 0 - Max 4397 - 32.5789773067481
Changed_Credit_Limit : Min -6.44 - Max 36.29 - 10.788655728870394
Num_Credit_Inquiries : Min 0.0 - Max 2597.0 - 28.006347495433996
Outstanding_Debt : Min 0.23 - Max 4998.07 - 1510.6508349799412
Credit_Utilization_Ratio : Min 20.88125003902868 - Max 49.56451934738699 - 32.22058395632018
Credit_History_Age : Min 0.0 - Max 404.0 - 193.59627533758496
Total_EMI_per_month : Min 4.4628374669131645 - Max 82204.0 - 1403.5067912154855
Amount_invested_monthly : Min 0.0 - Max 10000.0 - 614.0763433835015
Monthly_Balance : Min 0.4534564914083034 - Max 1552.9460937445635 - 381.59373833367243
Count_Auto Loan : Min 0.0 - Max 4.0 - 0.42262207718331785
Count_Credit-Builder Loan : Min 0.0 - Max 4.0 - 0.4532934131736527
Count_Personal Loan : Min 0.0 - Max 4.0 - 0.4406287425149701
Count_Home Equity Loan : Min 0.0 - Max 5.0 - 0.44211251160144904
Count_Not Specified : Min 0.0 - Max 4.0 - 0.44275106293790045
Count_Mortgage Loan : Min 0 - Max 5 - 0.43859649122807015
Count_Student Loan : Min 0 - Max 5 - 0.4400335309262918
Count_Debt Consolidation Loan : Min 0 - Max 5 - 0.43359679061134065
Count_Payday Loan : Min 0 - Max 5 - 0.4569187473803964
```

```
# Dropping values
```

```
# Age
df.drop(df[df.Age < 0].index, inplace=True)

# Num_Bank_Accounts
df.drop(df[df["Num_Bank_Accounts"] < 0].index, inplace=True)

# Reseting index values
df.reset_index(drop=True, inplace=True)

# Checking Null Values
#features_null_values = []
#for col in numerical_data:
#    if df[col].isnull().sum() !=0:
#        features_null_values.append(col)

#####
#print(features_null_values)

df.isna().sum()
```

```
Month                       2
Age                         2
Occupation                  2
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           1
Num_Credit_Card             1
Interest_Rate               1
Delay_from_due_date         3
Num_of_Delayed_Payment      0
Changed_Credit_Limit        1
Num_Credit_Inquiries        3
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          3
Payment_of_Min_Amount       0
Total_EMI_per_month         4
Amount_invested_monthly     0
Payment_Behaviour           1
Monthly_Balance             1
```

```
Monthly_Balance                1
Credit_Score                   0
Count_Auto Loan                1
Count_Credit-Builder Loan      2
Count_Personal Loan            2
Count_Home Equity Loan         1
Count_Not Specified            4
Count_Mortgage Loan            0
Count_Student Loan             0
Count_Debt Consolidation Loan  0
Count_Payday Loan              0
dtype: int64
```

## Splitting the data

```
X = df.drop('Credit_Score', axis=1)
y = df['Credit_Score']
```

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33397 entries, 0 to 33396
Data columns (total 30 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Month                          33395 non-null  object 
 1   Age                            33395 non-null  float64
 2   Occupation                     33395 non-null  object 
 3   Annual_Income                  33397 non-null  float64
 4   Monthly_Inhand_Salary          33397 non-null  float64
 5   Num_Bank_Accounts              33396 non-null  float64
 6   Num_Credit_Card                33396 non-null  float64
 7   Interest_Rate                  33396 non-null  float64
 8   Delay_from_due_date            33394 non-null  float64
 9   Num_of_Delayed_Payment         33397 non-null  int64  
 10  Changed_Credit_Limit           33396 non-null  float64
 11  Num_Credit_Inquiries           33394 non-null  float64
 12  Credit_Mix                     33397 non-null  object 
```

```
 13  Outstanding_Debt              33397 non-null  float64
 14  Credit_Utilization_Ratio      33397 non-null  float64
 15  Credit_History_Age            33394 non-null  float64
 16  Payment_of_Min_Amount         33397 non-null  object
 17  Total_EMI_per_month           33393 non-null  float64
 18  Amount_invested_monthly       33397 non-null  float64
 19  Payment_Behaviour             33396 non-null  object
 20  Monthly_Balance               33396 non-null  float64
 21  Count_Auto Loan               33396 non-null  float64
 22  Count_Credit-Builder Loan     33395 non-null  float64
 23  Count_Personal Loan           33395 non-null  float64
 24  Count_Home Equity Loan        33396 non-null  float64
 25  Count_Not Specified           33393 non-null  float64
 26  Count_Mortgage Loan           33397 non-null  int64
 27  Count_Student Loan            33397 non-null  int64
 28  Count_Debt Consolidation Loan  33397 non-null  int64
 29  Count_Payday Loan             33397 non-null  int64
dtypes: float64(20), int64(5), object(5)
memory usage: 7.6+ MB
```

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

# Selecting again the numerical and categorical data but this time from splitted dataset
numerical_format = ['float64', 'int']
numerical_data = X.select_dtypes(include=numerical_format).columns
categorical_data = X.select_dtypes(include='object').columns
```

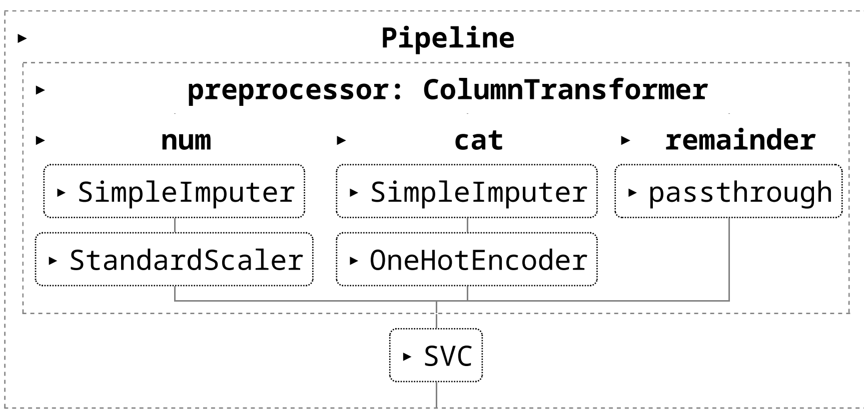
## Replacing NaN Values from Numerical and Categorical Columns with Mean and most frequent words.

```
# Numerical pipeline
mean_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
```

```
])

# Categorical pipeline
cat_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine pipelines using ColumnTransformer
preprocessor = ColumnTransformer([
    ('num', mean_pipeline, numerical_data),
    ('cat', cat_pipeline, categorical_data)
], remainder='passthrough')
```

## Applying pipelines and creating a SVC model

```
# Create a complete pipeline with the preprocessor and the SVM model
svm_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', SVC(kernel='rbf', C=10))
])

# Fit the SVM model on the training data
svm_pipeline.fit(X_train, y_train)
```



```
# Predict the target values for the test data
y_pred = svm_pipeline.predict(X_test)

# Evaluate and report the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the SVM model: {accuracy:.2f}")
```

```
Accuracy of the SVM model: 0.69
```

```
import pickle
```

```
with open('svm_model.pkl','wb') as file:
  pickle.dump(svm_pipeline, file)
```

```
# Save the trained model to a file
#model_filename = 'svm_model.joblib'
#model_filename = 'svm_model.pkl'
#joblib.dump(svm_pipeline, model_filename)
#print(f"Model saved to {model_filename}")
```

```
df.iloc[0]
```