Credit Score Classification

Banks and credit card companies rely on credit scores to assess the creditworthiness of individuals. The use of Machine Learning algorithms has become prevalent in this process, aiding institutions in categorizing customers based on their credit histories for swift loan approvals. If you are interested in leveraging Machine Learning for credit score classification, this article is tailored for you. Within this guide, I'll walk you through the process of credit score classification using Machine Learning techniques in Python.

Credit Score Classification by Banks and Credit Card Companies

Banks and credit card companies categorize their customers into three main credit scores: Good, Standard, and Poor.

- **Good Credit Score:** Individuals with a good credit score are typically eligible for loans from various banks and financial institutions.

```
# Importing Libraries
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"
```

Dataset Attributes:

- **ID:** Unique ID of the record
- **Customer_ID:** Unique ID of the customer
- **Month:** Month of the year
- **Name:** The name of the person
- **Age:** The age of the person
- **SSN:** Social Security Number of the person
- **Occupation:** The occupation of the person
- **Annual_Income:** The Annual Income of the person
- **Monthly_Inhand_Salary:** Monthly in-hand salary of the person
- **Num_Bank_Accounts:** The number of bank accounts of the person
- **Num_Credit_Card:** Number of credit cards the person is having
- **Interest_Rate:** The interest rate on the credit card of the person
- **Num_of_Loan:** The number of loans taken by the person from the bank
- **Type_of_Loan:** The types of loans taken by the person from the bank
- **Delay_from_due_date:** The average number of days delayed by the person from the date of payment
- **Num_of_Delayed_Payment:** Number of payments delayed by the person
- **Changed_Credit_Card:** The percentage change in the credit card limit of the person
- **Num_Credit_Inquiries:** The number of credit card inquiries by the person
- **Credit_Mix:** Classification of Credit Mix of the customer
- **Outstanding_Debt:** The outstanding balance of the person
- **Credit_Utilization_Ratio:** The credit utilization ratio of the credit card of the customer
- **Credit_History_Age:** The age of the credit history of the person
- **Payment_of_Min_Amount:** Yes if the person paid the minimum amount to be paid only, otherwise no.
- **Total_EMI_per_month:** The total EMI per month of the person
- **Amount_invested_monthly:** The monthly amount invested by the person
- **Payment_Behaviour:** The payment behaviour of the person
- **Monthly_Balance:** The monthly balance left in the account of the person
- **Credit_Score:** The credit score of the person

The Credit_Score column is the target variable in this problem. You are required to find relationships based on how banks classify credit scores and train a model to classify the credit score of a person.

```
# Reading the Data
data = pd.read_csv("Dataset.csv")
print(data.head())
```

```
     ID  Customer_ID  Month           Name   Age          SSN Occupation  \
0  5634         3392      1  Aaron Maashoh  23.0  821000265.0  Scientist   
1  5635         3392      2  Aaron Maashoh  23.0  821000265.0  Scientist   
2  5636         3392      3  Aaron Maashoh  23.0  821000265.0  Scientist   
3  5637         3392      4  Aaron Maashoh  23.0  821000265.0  Scientist   
4  5638         3392      5  Aaron Maashoh  23.0  821000265.0  Scientist   

   Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  ...  Credit_Mix  \
0       19114.12            1824.843333                3.0  ...        Good   
1       19114.12            1824.843333                3.0  ...        Good   
2       19114.12            1824.843333                3.0  ...        Good   
3       19114.12            1824.843333                3.0  ...        Good   
4       19114.12            1824.843333                3.0  ...        Good   

   Outstanding_Debt  Credit_Utilization_Ratio Credit_History_Age  \
0            809.98                 26.822620              265.0   
1            809.98                 31.944960              266.0   
2            809.98                 28.609352              267.0   
3            809.98                 31.377862              268.0   
4            809.98                 24.797347              269.0   

   Payment_of_Min_Amount  Total_EMI_per_month  Amount_invested_monthly  \
0                     No            49.574949                 21.46538   
1                     No            49.574949                 21.46538   
2                     No            49.574949                 21.46538   
3                     No            49.574949                 21.46538   
4                     No            49.574949                 21.46538   

                 Payment_Behaviour Monthly_Balance  Credit_Score  
0   High_spent_Small_value_payments      312.494089          Good  
1    Low_spent_Large_value_payments      284.629162          Good  
2   Low_spent_Medium_value_payments      331.209863          Good  
3    Low_spent_Small_value_payments      223.451310          Good  
4  High_spent_Medium_value_payments      341.489231          Good  

[5 rows x 28 columns]
```

```
# Information about the Dataset
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        100000 non-null  int64  
 1   Customer_ID               100000 non-null  int64  
 2   Month                     100000 non-null  int64  
 3   Name                      100000 non-null  object 
 4   Age                       100000 non-null  float64
 5   SSN                       100000 non-null  float64
 6   Occupation                100000 non-null  object 
 7   Annual_Income             100000 non-null  float64
 8   Monthly_Inhand_Salary     100000 non-null  float64
 9   Num_Bank_Accounts         100000 non-null  float64
 10  Num_Credit_Card           100000 non-null  float64
 11  Interest_Rate             100000 non-null  float64
 12  Num_of_Loan               100000 non-null  float64
 13  Type_of_Loan              100000 non-null  object 
 14  Delay_from_due_date       100000 non-null  float64
 15  Num_of_Delayed_Payment    100000 non-null  float64
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  float64
 18  Credit_Mix                100000 non-null  object 
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        100000 non-null  float64
 22  Payment_of_Min_Amount     100000 non-null  object 
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object 
 26  Monthly_Balance           100000 non-null  float64
 27  Credit_Score              100000 non-null  object 
dtypes: float64(18), int64(3), object(7)
memory usage: 21.4+ MB
None
```

```
# Checking for null values
print(data.isnull().sum())
```

```
ID                          0
Customer_ID                 0
```

```
Month                       0
Name                        0
Age                         0
SSN                         0
Occupation                  0
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
Type_of_Loan                0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Credit_Score                0
dtype: int64
```

So the dataset doesnt have any null values. Now lets check Credit_Score column values:

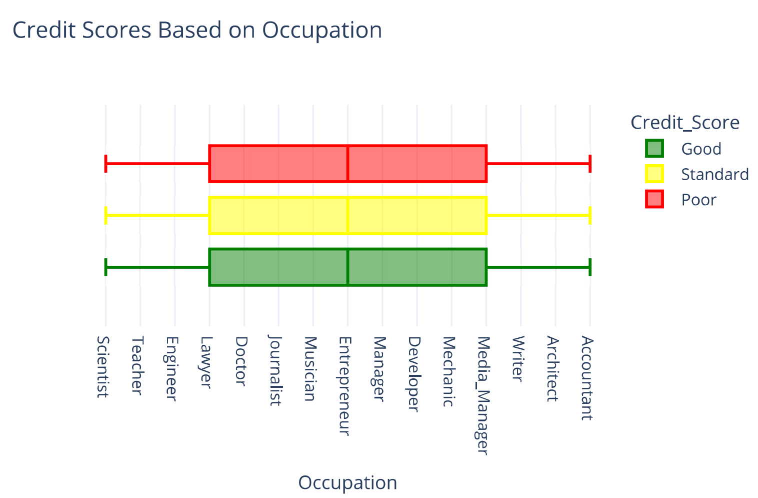
```
data["Credit_Score"].value_counts()
```

```
Standard    53174
Poor        28998
Good        17828
Name: Credit_Score, dtype: int64
```

Exploratory Data Analysis - Lets get to know more about this data

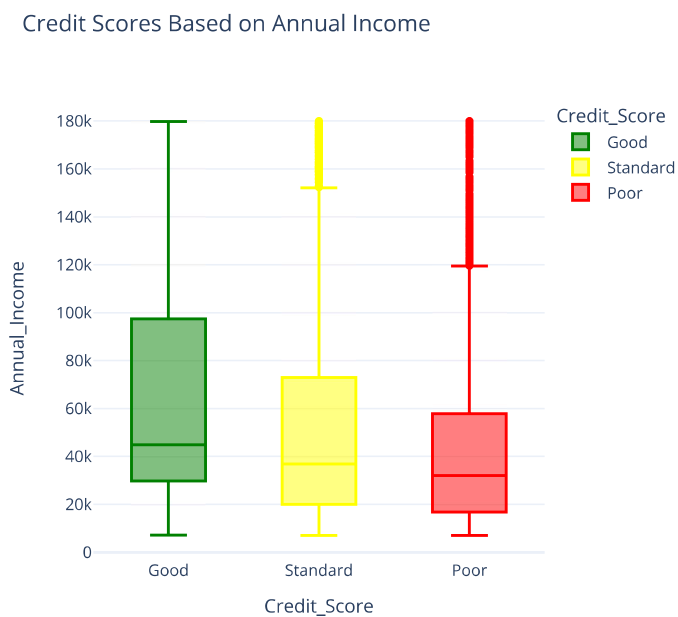
let's start by looking at the occupation of individuals to see if it has any influence on their credit scores.

```
fig = px.box(data,
             x="Occupation",
             color="Credit_Score",
             title="Credit Scores Based on Occupation",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_layout(height=400, width=600)
fig.show()
```



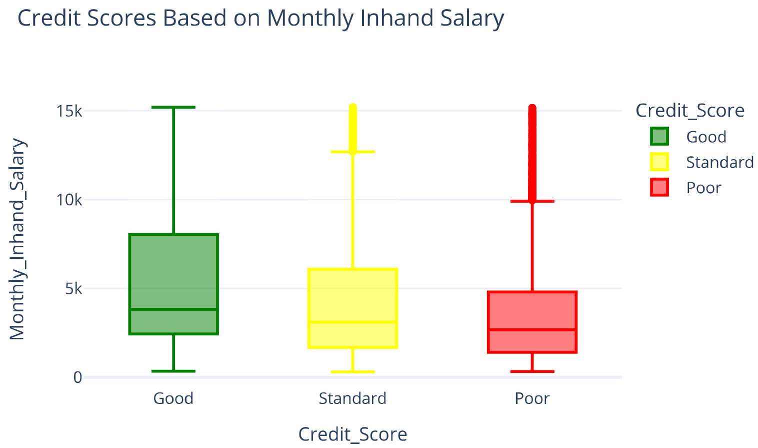
There's not much difference in the credit scores of all occupations mentioned in the data. Let's investigate whether there's a discernible impact of an individual's Annual Income on their credit scores.

```
# Define a custom color map for Credit_Score categories
color_map = {'Poor': 'red', 'Standard': 'yellow', 'Good': 'green'}

# Create the box plot using Plotly with custom color map
fig = px.box(data,
             x="Credit_Score",
             y="Annual_Income",
             color="Credit_Score",
             title="Credit Scores Based on Annual Income",
             color_discrete_map=color_map)
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



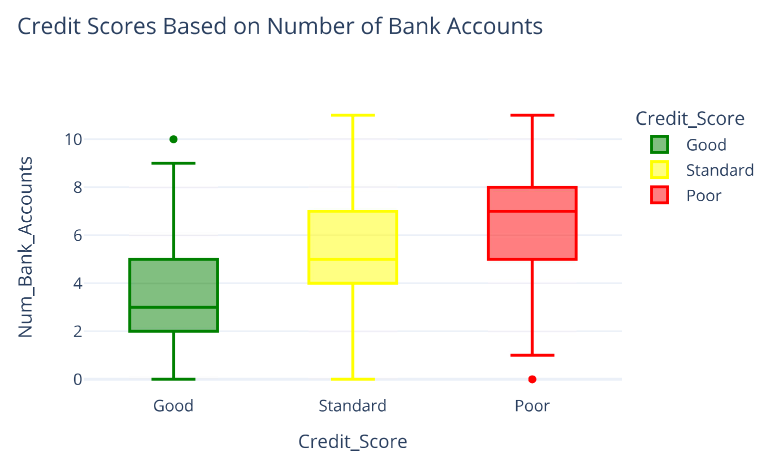
Based on the previous visualization, it suggests a positive correlation between higher annual earnings and better credit scores. Now, let's investigate whether there exists a relationship between the monthly in-hand salary and credit scores.

```
# Create the box plot to visualize Credit Scores based on Monthly Inhand Salary
fig = px.box(data,
             x="Credit_Score",
             y="Monthly_Inhand_Salary",
             color="Credit_Score",
             title="Credit Scores Based on Monthly Inhand Salary",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



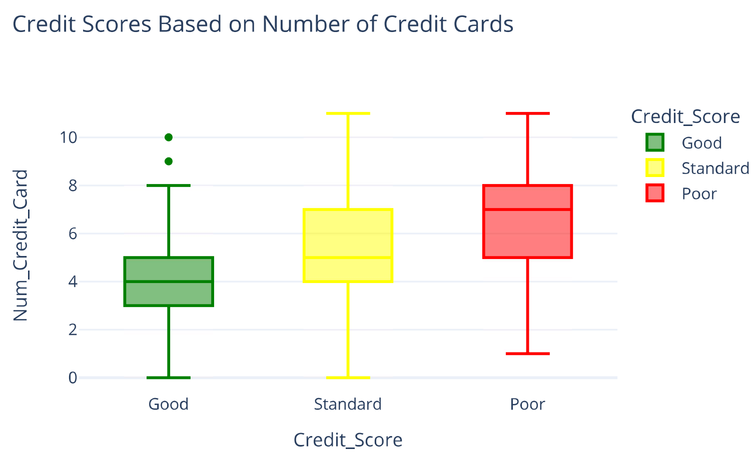
Similar to the pattern observed with annual income, a higher monthly in-hand salary appears to correspond to better credit scores. Now, let's investigate whether the number of bank accounts influences credit scores.

```
# Create a box plot to explore the impact of the Number of Bank Accounts on Credit Scores
fig = px.box(data,
             x="Credit_Score",
             y="Num_Bank_Accounts",
             color="Credit_Score",
             title="Credit Scores Based on Number of Bank Accounts",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



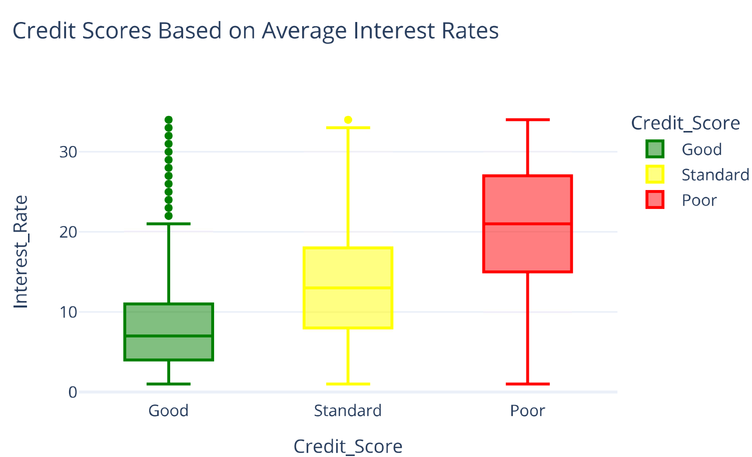
Having more than five accounts doesn't help in getting a good credit score. It seems having 2 to 3 bank accounts works best. Now, let's check if having more credit cards affects credit scores.

```
# Creating a box plot to explore the impact of the Number of Credit Cards on Credit Scores
fig = px.box(data,
             x="Credit_Score",
             y="Num_Credit_Card",
             color="Credit_Score",
             title="Credit Scores Based on Number of Credit Cards",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



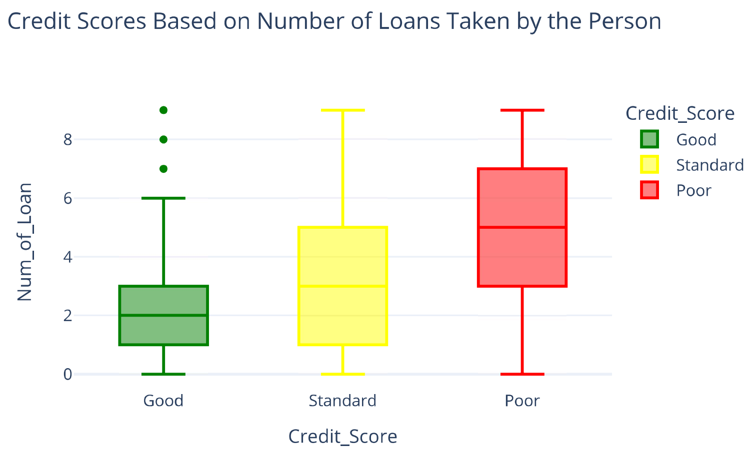
Similar to the impact observed with the number of bank accounts, having an abundance of credit cards doesn't seem to improve credit scores. Maintaining 3 to 5 credit cards is beneficial for your credit score. Now, let's explore how the average interest paid on loans and EMIs affects credit scores.

```
# Creating a box plot to explore the impact of Average Interest Rates on Credit Scores
fig = px.box(data,
             x="Credit_Score",
             y="Interest_Rate",
             color="Credit_Score",
             title="Credit Scores Based on Average Interest Rates",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



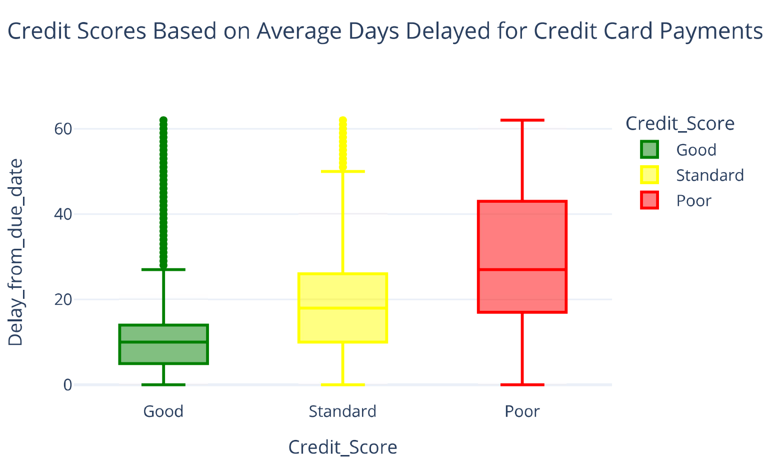
A credit score tends to be good when the average interest rate ranges between 4% and 11%. However, having an average interest rate exceeding 15% negatively affects credit scores. Now, let's examine the number of concurrent loans that contribute to a favorable credit score.

```
fig = px.box(data,
             x="Credit_Score",
             y="Num_of_Loan",
             color="Credit_Score",
             title="Credit Scores Based on Number of Loans Taken by the Person",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



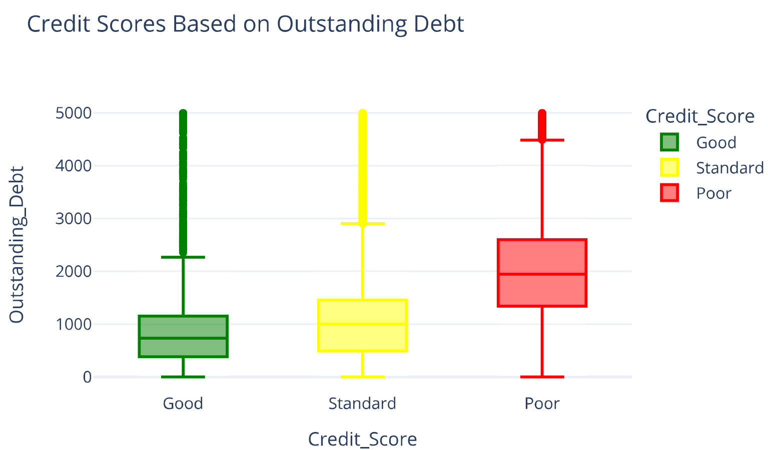
Maintaining a good credit score typically involves managing 1 to 3 loans simultaneously. However, having more than three concurrent loans tends to adversely affect your credit scores. Now, let's investigate whether delaying payments on the due date has an impact on credit scores.

```
# Create a box plot to explore the impact of Average Delay from Due Date on Credit Scores
fig = px.box(data,
             x="Credit_Score",
             y="Delay_from_due_date",
             color="Credit_Score",
             title="Credit Scores Based on Average Days Delayed for Credit Card Payments",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



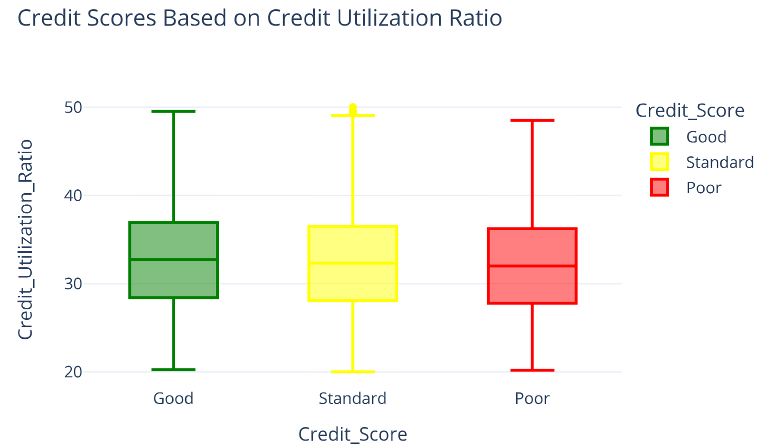
Delaying your credit card payment by 5 to 14 days from the due date appears acceptable without significantly impacting your credit scores. However, prolonging payment delays beyond 17 days from the due date is likely to have a negative effect on your credit scores. Now, let's explore whether frequently delaying payments affects

```
import plotly.express as px

# Create a box plot to explore the impact of Outstanding Debt on Credit Scores
fig = px.box(data,
             x="Credit_Score",
             y="Outstanding_Debt",
             color="Credit_Score",
             title="Credit Scores Based on Outstanding Debt",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



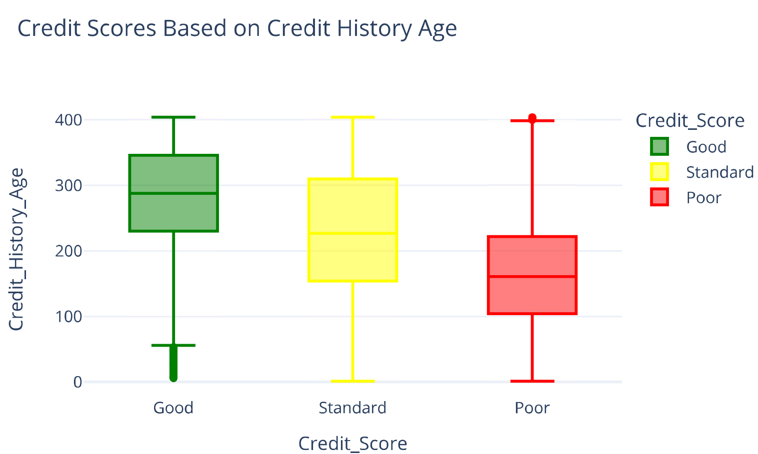
Having an outstanding debt within the range of $380-1150$ seems to have no significant impact on your credit scores. However, consistently maintaining a debt exceeding $1338 is likely to negatively affect your credit scores. Now, let's investigate whether a high credit utilization ratio influences credit scores.

```
fig = px.box(data,
             x="Credit_Score",
             y="Credit_Utilization_Ratio",
             color="Credit_Score",
             title="Credit Scores Based on Credit Utilization Ratio",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



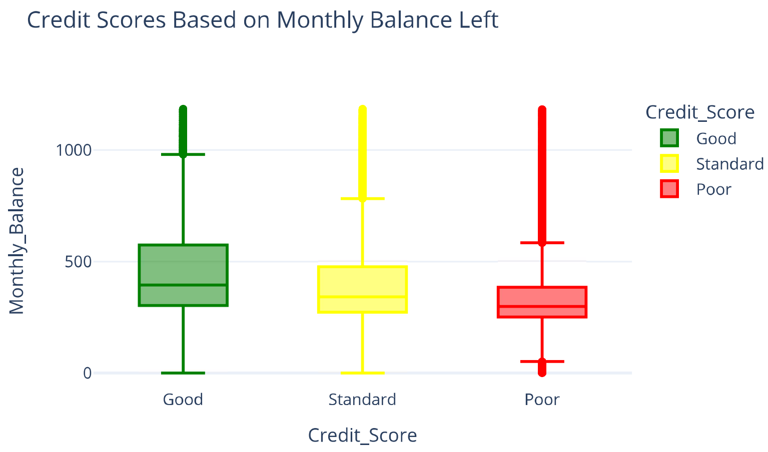
As per the previous visualization, the credit utilization ratio, calculated by dividing total debt by total available credit, doesn't seem to have an impact on credit scores. Now, let's explore how a person's credit history age influences their credit scores.

```
fig = px.box(data,
             x="Credit_Score",
             y="Credit_History_Age",
             color="Credit_Score",
             title="Credit Scores Based on Credit History Age",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



let's see if having a low amount at the end of the month affects credit scores or not:

```
# Creating a box plot to visualize Credit Scores based on Monthly Balance
fig = px.box(data,
             x="Credit_Score",
             y="Monthly_Balance",
             color="Credit_Score",
             title="Credit Scores Based on Monthly Balance Left",
             color_discrete_map={'Poor':'red', 'Standard':'yellow', 'Good':'green'})

fig.update_traces(quartilemethod="exclusive")
fig.update_layout(height=400, width=600)
fig.show()
```



Having a substantial monthly balance in your account by the end of the month seems beneficial for your credit scores. Conversely, maintaining a monthly balance below $250 is detrimental to your credit scores.

Credit Score Classification Model

One more important feature (Credit Mix) in the dataset is valuable for determining credit scores. The credit mix feature tells about the types of credits and loans you have taken.

As the Credit_Mix column is categorical, I will transform it into a numerical feature so that we can use it to train a Machine Learning model for the task of credit score classification:

```
data["Credit_Mix"].replace({"Standard": 1, "Good": 2, "Bad": 0}, inplace=True)
```

Now I will split the data into features and labels by selecting the features we found important for our model:

```
from sklearn.model_selection import train_test_split
x = np.array(data[["Annual_Income", "Monthly_Inhand_Salary",
                   "Num_Bank_Accounts", "Num_Credit_Card",
                   "Interest_Rate", "Num_of_Loan",
                   "Delay_from_due_date", "Num_of_Delayed_Payment",
                   "Credit_Mix", "Outstanding_Debt",
                   "Credit_History_Age", "Monthly_Balance"]])
y = np.array(data[["Credit_Score"]])
```

Now, let's split the data into training and test sets and proceed further by training a credit score classification model:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Splitting the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

# Creating and training the Random Forest Classifier model
model = RandomForestClassifier(random_state=42)
model.fit(x_train, y_train)

# Predicting on the test set
y_pred = model.predict(x_test)

# Assessing model performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(report)
```

```
print(report)
```

```
<ipython-input-23-78046ee9d5a5>:10: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

Accuracy: 0.81
Classification Report:
              precision    recall  f1-score   support

        Good       0.77      0.76      0.77      5866
        Poor       0.79      0.83      0.81      9633
    Standard       0.83      0.81      0.82     17501

    accuracy                           0.81     33000
   macro avg       0.80      0.80      0.80     33000
weighted avg       0.81      0.81      0.81     33000
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.33,
                                                random_state=42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(xtrain, ytrain)
```

```
<ipython-input-24-ea8253e80d41>:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

▾ RandomForestClassifier
RandomForestClassifier()
```

Now, let's make predictions from our model by giving inputs to our model according to the features we used to train the model:

```
print("Credit Score Prediction : ")
a = float(input("Annual Income: "))
b = float(input("Monthly Inhand Salary: "))
c = float(input("Number of Bank Accounts: "))
d = float(input("Number of Credit cards: "))
e = float(input("Interest rate: "))
f = float(input("Number of Loans: "))
g = float(input("Average number of days delayed by the person: "))
h = float(input("Number of delayed payments: "))
i = input("Credit Mix (Bad: 0, Standard: 1, Good: 3) : ")
j = float(input("Outstanding Debt: "))
k = float(input("Credit History Age: "))
l = float(input("Monthly Balance: "))
```