classification-using-random-forest

June 16, 2024

[26]:
```
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"
```

[27]:
```
data = pd.read_csv("C:/Users/suova/OneDrive/Desktop/Credit Score Data/train.
↪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]
```

[28]:
```python
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
```

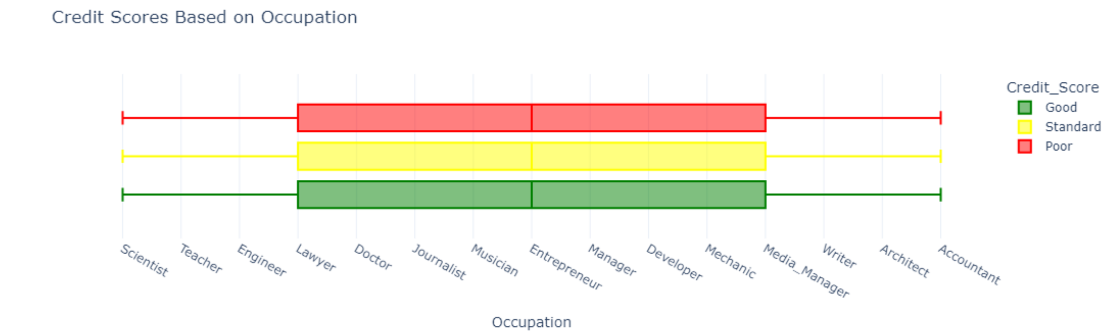
[29]:
```
print(data.isnull().sum())
```

```
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
```

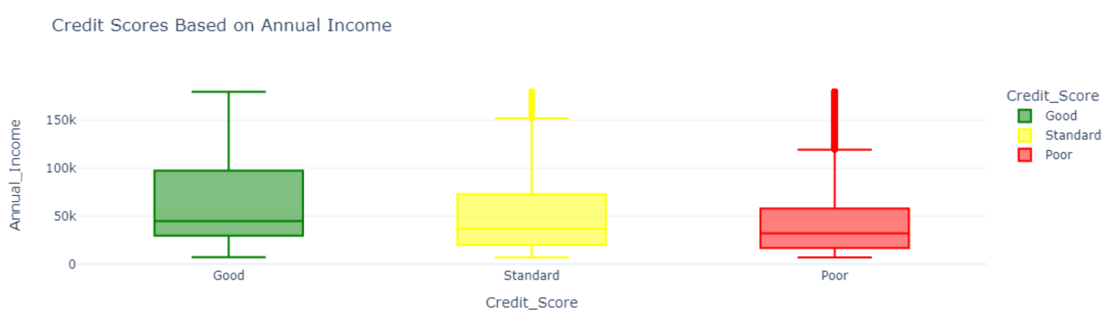
[30]:
```
data["Credit_Score"].value_counts()
```

[30]:
```
Standard    53174
Poor        28998
Good        17828
Name: Credit_Score, dtype: int64
```

[31]:
```
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.show()
```

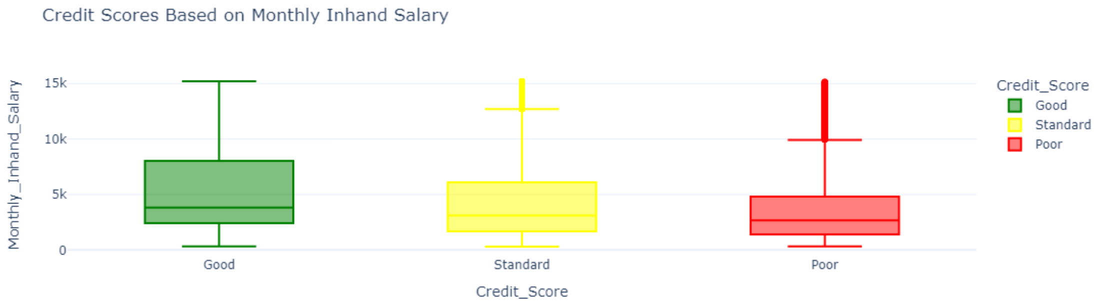


[32]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Annual_Income",
             color="Credit_Score",
             title="Credit Scores Based on Annual Income",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

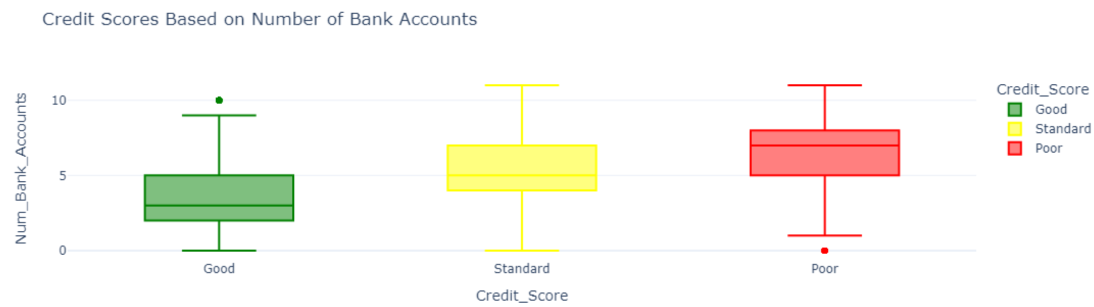


[33]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Monthly_Inhand_Salary",
             color="Credit_Score",
             title="Credit Scores Based on Monthly Inhand Salary",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
```

```
fig.show()
```

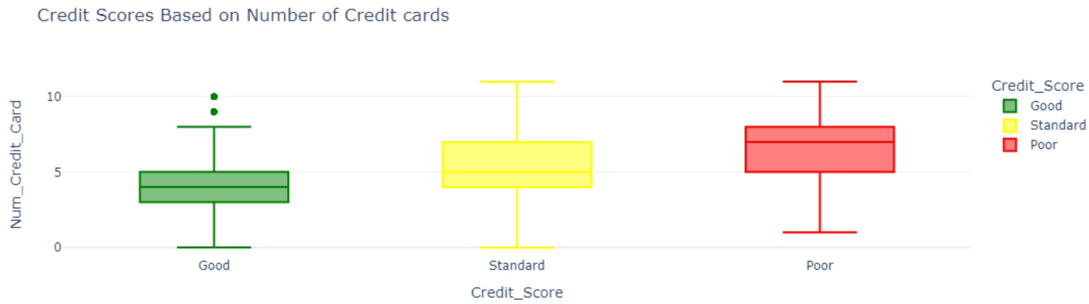


[34]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Num_Bank_Accounts",
             color="Credit_Score",
             title="Credit Scores Based on Number of Bank Accounts",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

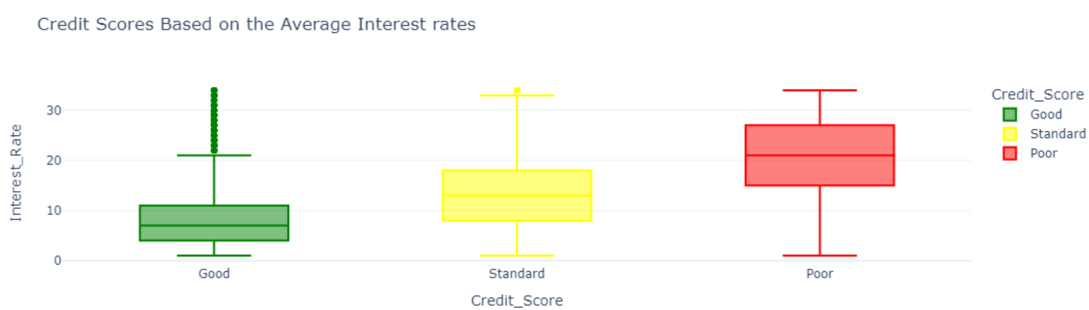


[35]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Num_Credit_Card",
             color="Credit_Score",
             title="Credit Scores Based on Number of Credit cards",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
```

```
                                  'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

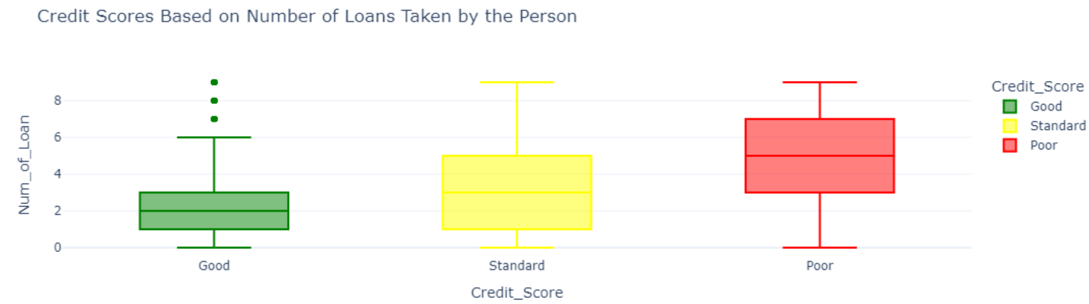


[36]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Interest_Rate",
             color="Credit_Score",
             title="Credit Scores Based on the Average Interest rates",
             color_discrete_map={'Poor':'red',
                                  'Standard':'yellow',
                                  'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

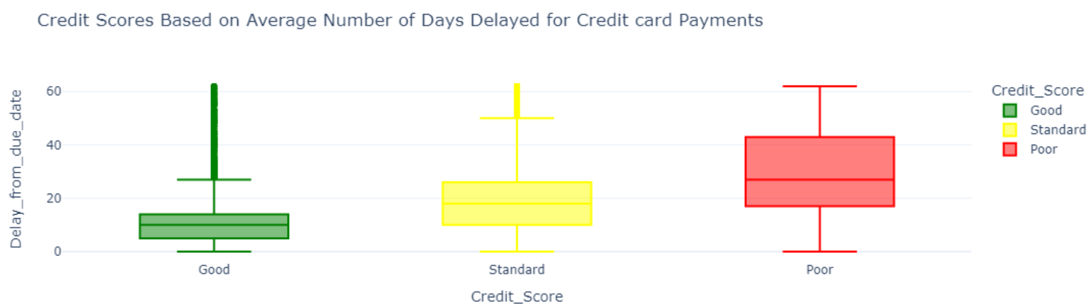


[37]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Num_of_Loan",
             color="Credit_Score",
             title="Credit Scores Based on Number of Loans Taken by the Person",
```

```
                 color_discrete_map={'Poor':'red',
                                     'Standard':'yellow',
                                     'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

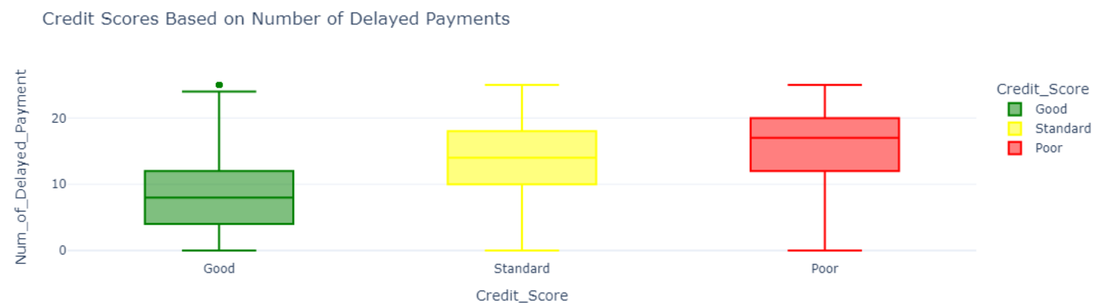


[38]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Delay_from_due_date",
             color="Credit_Score",
             title="Credit Scores Based on Average Number of Days Delayed for␣
 ↪Credit card Payments",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

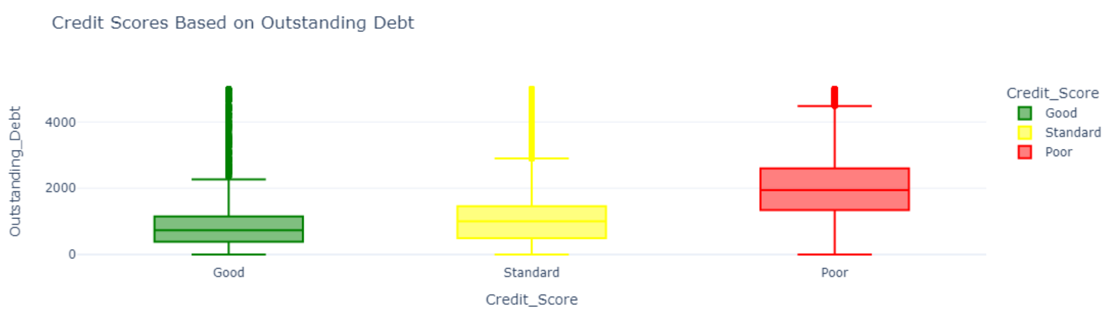


[39]:
```
fig = px.box(data,
             x="Credit_Score",
```

```
             y="Num_of_Delayed_Payment",
             color="Credit_Score",
             title="Credit Scores Based on Number of Delayed Payments",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

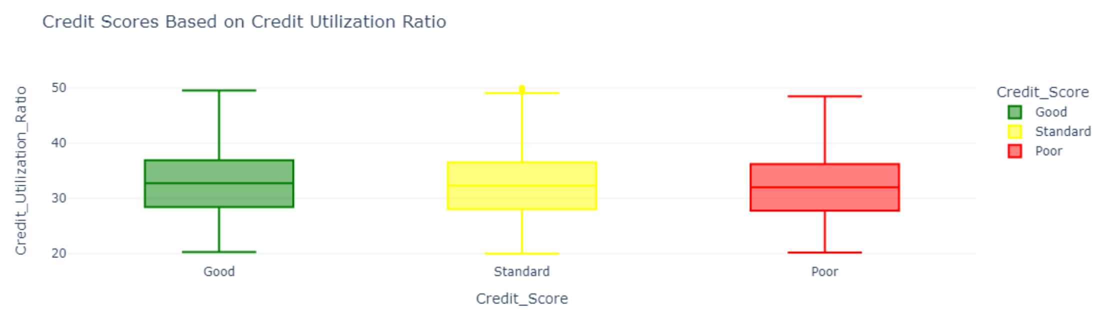


[40]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Outstanding_Debt",
             color="Credit_Score",
             title="Credit Scores Based on Outstanding Debt",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



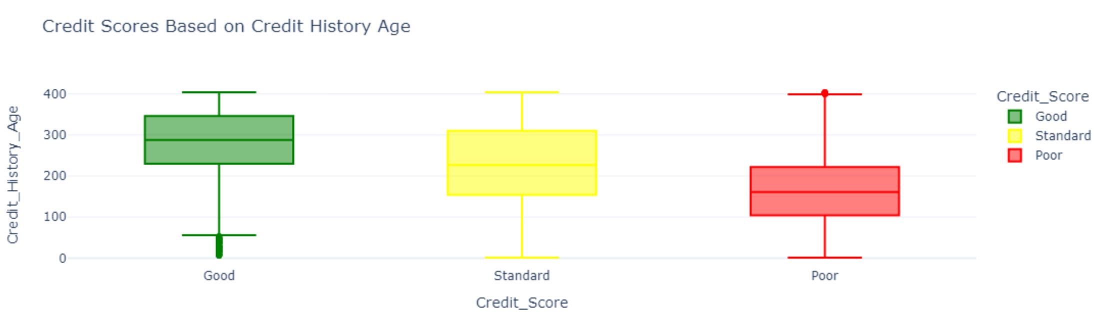
[41]:

```
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.show()
```

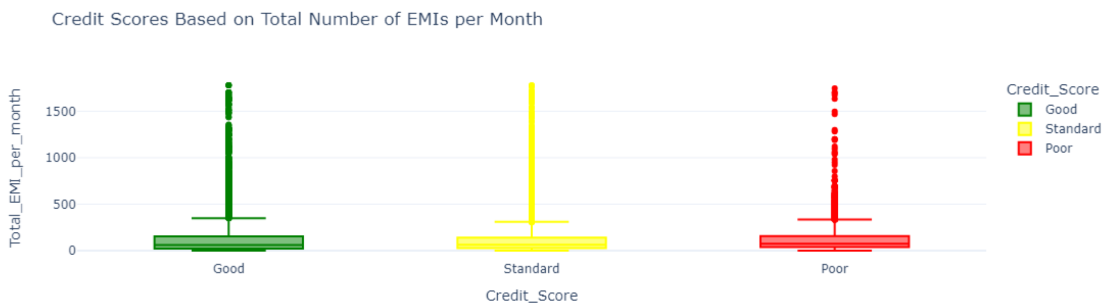


[42]:

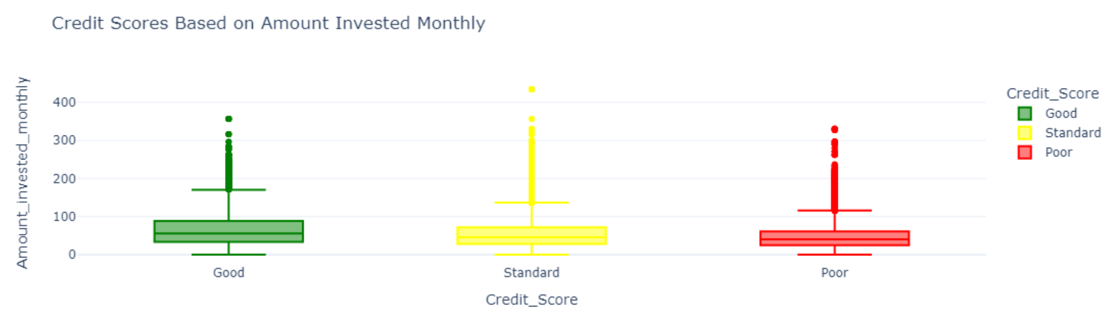
```
fig = px.box(data,
             x="Credit_Score",
             y="Credit_History_Age",
             color="Credit_Score",
             title="Credit Scores Based on Credit History Age",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
         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



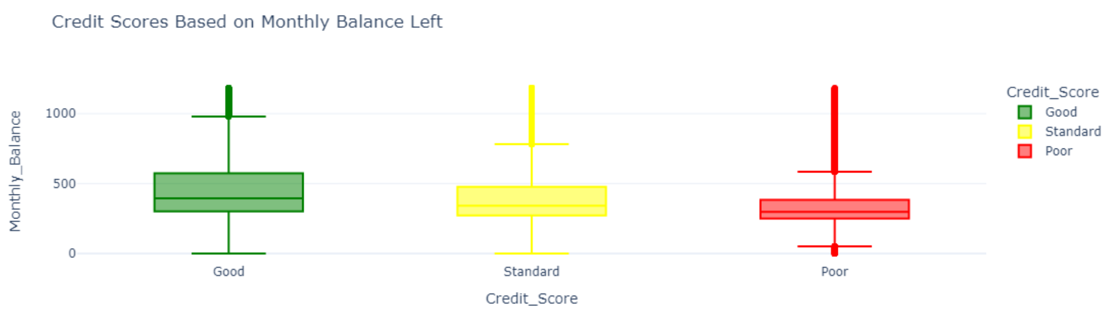
[43]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Total_EMI_per_month",
             color="Credit_Score",
             title="Credit Scores Based on Total Number of EMIs per Month",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



[44]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Amount_invested_monthly",
             color="Credit_Score",
             title="Credit Scores Based on Amount Invested Monthly",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



[45]:
```
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.show()
```



[46]:
```
data["Credit_Mix"] = data["Credit_Mix"].map({"Standard": 1,
                               "Good": 2,
                               "Bad": 0})
```

[47]:
```
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"]])
```

[48]:
```
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)
```

```
C:\Users\suova\AppData\Local\Temp\ipykernel_6664\2049170333.py: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().

[48]: RandomForestClassifier()

[49]:
```
#Accuracy of the model
from sklearn.metrics import accuracy_score,precision_score,recall_score
print(accuracy_score(ytest, model.predict(xtest)))
```

0.807

[50]:
```
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: 2) : ")
j = float(input("Outstanding Debt: "))
k = float(input("Credit History Age: "))
l = float(input("Monthly Balance: "))

features = np.array([[a, b, c, d, e, f, g, h, i, j, k, l]])
print("Predicted Credit Score = ", model.predict(features))
```

```
Credit Score Prediction :
Annual Income: 19114.12
Monthly Inhand Salary: 1824.8433
Number of Bank Accounts: 2
Number of Credit cards: 2
Interest rate: 9
Number of Loans: 2
Average number of days delayed by the person: 12
Number of delayed payments: 3
Credit Mix (Bad: 0, Standard: 1, Good: 2) : 2
Outstanding Debt: 250
Credit History Age: 200
Monthly Balance: 310
Predicted Credit Score =  ['Good']
```

Annual Income: 19114.12 Monthly Inhand Salary: 1824.843333 Number of Bank Accounts: 2 Number of Credit cards: 2 Interest rate: 9 Number of Loans: 2 Average number of days delayed by the person: 12 Number of delayed payments: 3 Credit Mix (Bad: 0, Standard: 1, Good: 2) : 2 Outstanding Debt: 250 Credit History Age: 200 Monthly Balance: 310 Predicted Credit Score = ['Good']

Credit Score Prediction : Annual Income: 350000 Monthly Inhand Salary: 18000 Number of Bank Accounts: 8 Number of Credit cards: 5 Interest rate: 9 Number of Loans: 9 Average number of days delayed by the person: 234 Number of delayed payments: 13 Credit Mix (Bad: 0, Standard: 1, Good: 2) : 0 Outstanding Debt: 789 Credit History Age: 123 Monthly Balance: 567 Predicted Credit Score = ['Standard']

Credit Score Prediction : Annual Income: 34081.38 Monthly Inhand Salary: 2611.115 Number of Bank Accounts: 8 Number of Credit cards: 7 Interest rate: 15 Number of Loans: 3 Average number of days delayed by the person: 30 Number of delayed payments: 14 Credit Mix (Bad: 0, Standard: 1, Good: 2) : 1 Outstanding Debt: 1704.18 Credit History Age: 176 Monthly Balance: 392.19 Predicted Credit Score = ['Poor']

[ ]:

```

```