Credit Score Analysis

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import time
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
dataset=pd.read_csv("Creditscore.csv")
```

In [3]:
```
dataset.isna().sum()
```

Out[3]:
```
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
```

In [4]:
```
#Finding Null values in the dataset
dataset.isnull().sum()
```

Out[4]:
```
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
```

In [5]:
```
dataset.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
```

In [6]:
```
dataset.head()
```

Out[6]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income
0	5634	3392	1	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
1	5635	3392	2	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
2	5636	3392	3	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
3	5637	3392	4	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
4	5638	3392	5	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12

5 rows × 28 columns

In [7]:
```
dataset['Credit_Score'].value_counts()
```

Out[7]:
```
Credit_Score
Standard    53174
Poor        28998
Good        17828
Name: count, dtype: int64
```

In [8]:
```
dataset.describe()
```

Out[8]:

	ID	Customer_ID	Month	Age	SSN	Annu
count	100000.000000	100000.000000	100000.000000	100000.000000	1.000000e+05	100
mean	80631.500000	25982.666640	4.500000	33.316340	5.004617e+08	50
std	43301.486619	14340.543051	2.291299	10.764812	2.908267e+08	38
min	5634.000000	1006.000000	1.000000	14.000000	8.134900e+04	7
25%	43132.750000	13664.500000	2.750000	24.000000	2.451686e+08	19
50%	80631.500000	25777.000000	4.500000	33.000000	5.006886e+08	36
75%	118130.250000	38385.000000	6.250000	42.000000	7.560027e+08	71
max	155629.000000	50999.000000	8.000000	56.000000	9.999934e+08	179

8 rows × 21 columns

In [9]:
```
dataset.corr(numeric_only=True)
```

Out[9]:

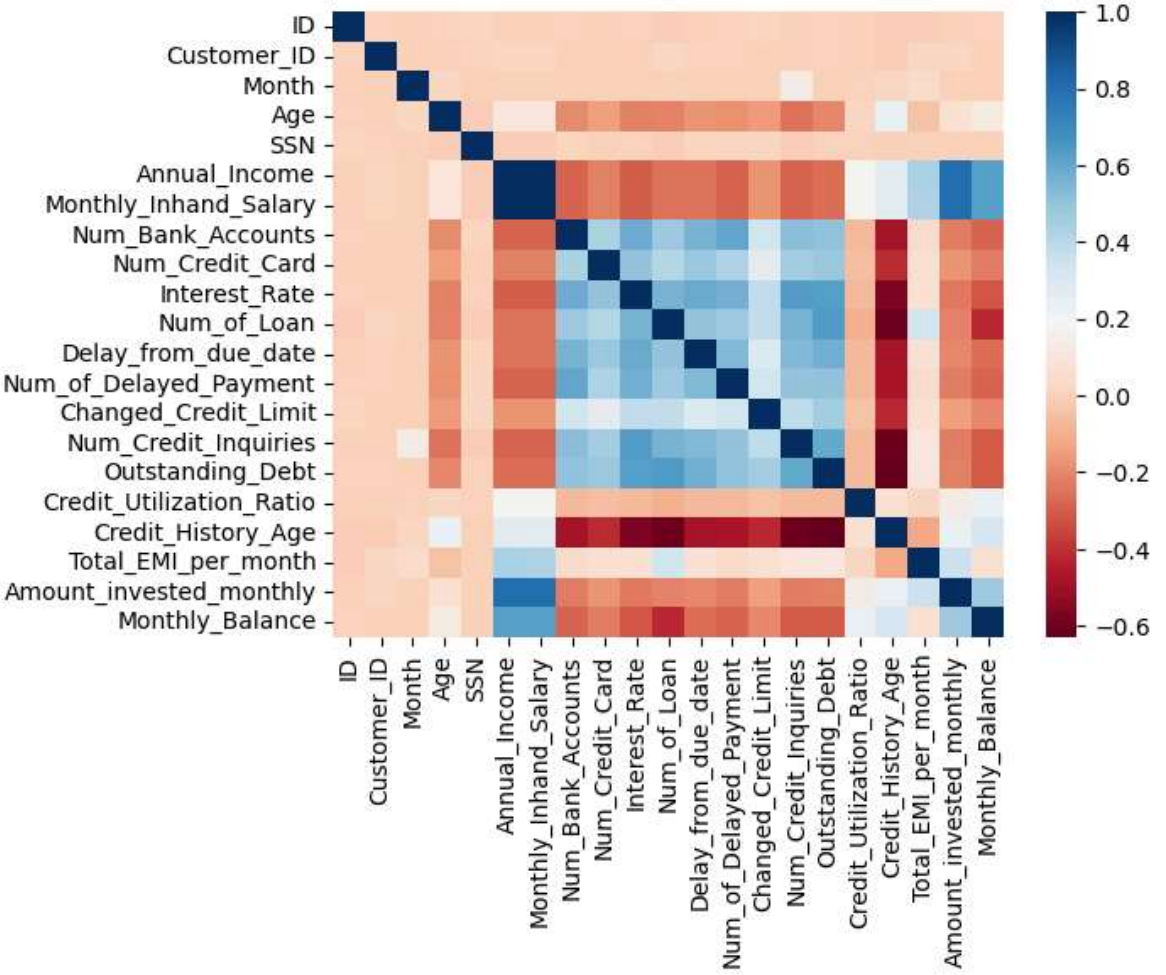
	ID	Customer_ID	Month	Age	S
ID	1.000000	1.235872e-03	5.291503e-05	0.003338	1.170712e-
Customer_ID	0.001236	1.000000e+00	-6.660113e-17	-0.002172	5.769305e-
Month	0.000053	-6.660113e-17	1.000000e+00	0.016990	-1.55749
Age	0.003338	-2.172258e-03	1.698955e-02	1.000000	-8.41099
SSN	0.011707	5.769305e-03	-1.557496e-16	-0.008411	1.000000e+
Annual_Income	-0.005357	1.076923e-02	1.530483e-17	0.091525	-8.04194
Monthly_Inhand_Salary	-0.004792	1.019558e-02	-1.818625e-03	0.090636	-8.36708
Num_Bank_Accounts	0.004826	-1.188701e-03	7.909795e-05	-0.190415	1.165117e-
Num_Credit_Card	-0.001900	-4.600690e-03	8.973261e-05	-0.148567	-4.96094
Interest_Rate	0.003957	-4.712091e-03	6.790115e-18	-0.217557	4.774004e-
Num_of_Loan	-0.010136	1.482832e-02	1.555542e-16	-0.213533	-1.08052
Delay_from_due_date	-0.004730	4.581213e-03	3.418202e-04	-0.174119	1.069729e-
Num_of_Delayed_Payment	0.001859	5.163216e-03	1.154567e-04	-0.184264	1.378698e-
Changed_Credit_Limit	0.008670	6.599255e-04	-6.708862e-04	-0.157254	1.975228e-
Num_Credit_Inquiries	-0.002201	3.516144e-04	1.396038e-01	-0.250960	-6.68437
Outstanding_Debt	0.002941	4.381594e-03	3.200578e-17	-0.202374	2.950511e-
Credit_Utilization_Ratio	-0.005402	-2.173680e-03	2.439924e-03	0.025492	2.652701e-
Credit_History_Age	-0.005604	-1.482226e-02	2.278441e-02	0.234618	-3.57204
Total_EMI_per_month	-0.009127	2.390433e-02	4.971583e-02	-0.047334	-4.69563
Amount_invested_monthly	-0.004044	1.632440e-02	-1.043065e-16	0.071045	-2.95190
Monthly_Balance	0.000859	-9.215332e-04	-2.094643e-03	0.116098	-2.72369

21 rows × 21 columns

Correlation

In [10]:
```
# Correlation is relation between  the two columns
# For this dataset we can see mixed corrrelation between independend and depende
# Fraud and cashout is 1.0 its a positive Correlation
```

In [11]:
```
sns.heatmap(dataset.corr(numeric_only=True),cmap='RdBu');
```



In [12]:
```
dataset.cov(numeric_only=True)
```

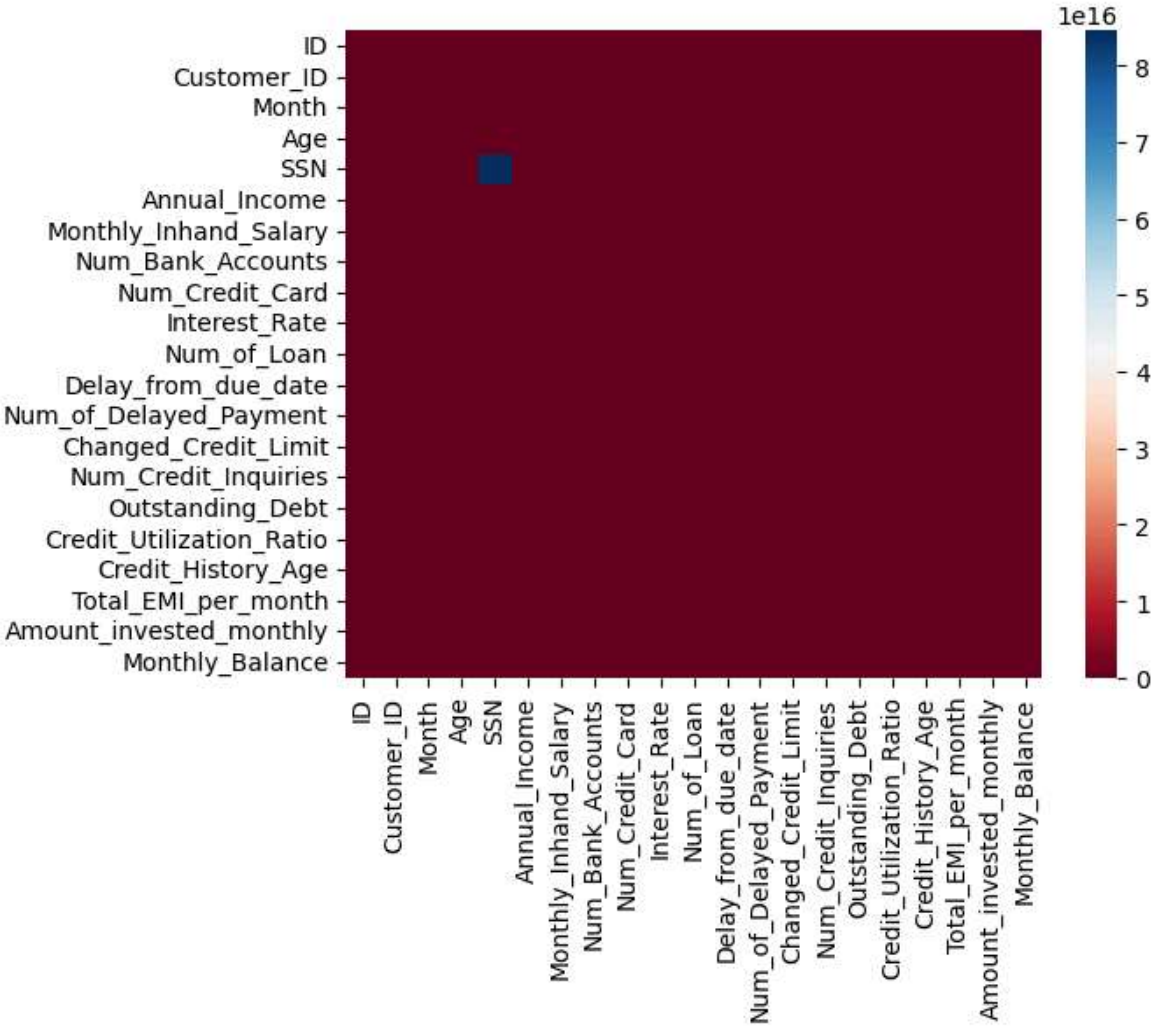
Out[12]:

	ID	Customer_ID	Month	Age
ID	1.875019e+09	7.674354e+05	5.250053e+00	1.555800e+03
Customer_ID	7.674354e+05	2.056512e+08	-7.163252e-14	-3.353386e+02
Month	5.250053e+00	-7.163252e-14	5.250053e+00	4.190542e-01
Age	1.555800e+03	-3.353386e+02	4.190542e-01	1.158812e+02
SSN	1.474305e+11	2.406154e+10	4.124683e-10	-2.633226e+07
Annual_Income	-8.884147e+06	5.914831e+06	-4.201908e-15	3.773430e+04
Monthly_Inhand_Salary	-6.611264e+05	4.658887e+05	-1.327791e+01	3.108941e+03
Num_Bank_Accounts	5.419494e+02	-4.420722e+01	4.700047e-04	-5.315736e+00
Num_Credit_Card	-1.701075e+02	-1.363797e+02	4.250043e-04	-3.305913e+00
Interest_Rate	1.497613e+03	-5.906862e+02	-8.952928e-18	-2.047185e+01
Num_of_Loan	-1.073725e+03	5.202081e+02	-2.389224e-18	-5.623297e+00
Delay_from_due_date	-3.032489e+03	9.726163e+02	1.159512e-02	-2.774907e+01
Num_of_Delayed_Payment	5.019506e+02	4.618205e+02	1.650017e-03	-1.237185e+01
Changed_Credit_Limit	2.481477e+03	6.255008e+01	-1.016010e-02	-1.118861e+01
Num_Credit_Inquiries	-3.685746e+02	1.950290e+01	1.237217e+00	-1.044907e+01
Outstanding_Debt	1.471178e+05	7.258189e+04	-1.432468e-16	-2.516470e+03
Credit_Utilization_Ratio	-1.196993e+03	-1.595020e+02	2.860638e-02	1.404178e+00
Credit_History_Age	-2.418860e+04	-2.118806e+04	5.203922e+00	2.517558e+02
Total_EMI_per_month	-5.227632e+04	4.534129e+04	1.506705e+01	-6.739525e+01
Amount_invested_monthly	-6.830735e+03	9.131551e+03	7.141026e-18	2.983202e+01
Monthly_Balance	7.498818e+03	-2.664898e+03	-9.678227e-01	2.520209e+02

21 rows × 21 columns

In [13]:
```
# Covariance
# Covariance is dfference between the two columns
# For this dataset we can see mixed covariance between independend and dependend
#covariance between newbalanceDest and oldbalanceDest is 1.33 its a positive cov
```

In [14]:
```
sns.heatmap(dataset.cov(numeric_only=True),cmap='RdBu');
```



In [15]:
```
indep=dataset[["Annual_Income","Num_Bank_Accounts","Monthly_Inhand_Salary","Inte

dep=dataset["Credit_Score"]
```

In [16]:
```
dataset['Credit_Score'].value_counts()
```

Out[16]:
```
Credit_Score
Standard    53174
Poor        28998
Good        17828
Name: count, dtype: int64
```

In [18]:
```
# using Selectkbest algorith using Chisquare or RMS formula,it will evaluate the
#Chi square will analyse the data and provide the result as per k value
#it provide Rscore value across all the models
def selectkbest(indep,dep,n):
        test = SelectKBest(score_func=chi2, k=n)
        fit1= test.fit(indep,dep)
        # summarize scores
        selectk_features = fit1.transform(indep)
        return selectk_features
```

In [19]:
```
# Using standard scaler and split the train and test set

def split_scalar(indep,dep):
        X_train, X_test, y_train, y_test = train_test_split(indep, dep, test_siz
        sc = StandardScaler()
```

```
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
        return X_train, X_test, y_train, y_test
```

In [20]:
```
#A confusion matrix is a table that is often used to evaluate the performance of
#classification algorithm on a set of labeled data
#It provides a summary of the predictions made by a classification model compare
#The matrix has four entries: true positive (TP), false positive (FP), true nega
def cm_prediction(classifier,X_test):
    y_pred = classifier.predict(X_test)
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
        #from sklearn.metrics import confusion_matrix
        #cm = confusion_matrix(y_test, y_pred)

    Accuracy=accuracy_score(y_test, y_pred )
#once above model creates in return  classifier,Accuracy,report,X_test,y_test,cm
    report=classification_report(y_test, y_pred)
    return  classifier,Accuracy,report,X_test,y_test,cm
```

In [21]:
```
# to create the Logistic regression model to find the score
def logistic(X_train,y_train,X_test):
        # Fitting K-NN to the Training set
        from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression(random_state = 0)
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [22]:
```
# to create the SVM Linear model to find the score
def svm_linear(X_train,y_train,X_test):


        from sklearn.svm import SVC
        classifier = SVC(kernel = 'linear', random_state = 0)
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [23]:
```
def svm_NL(X_train,y_train,X_test):

        from sklearn.svm import SVC
        classifier = SVC(kernel = 'rbf', random_state = 0)
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [24]:
```
def Navie(X_train,y_train,X_test):
        # Fitting K-NN to the Training set
        from sklearn.naive_bayes import GaussianNB
        classifier = GaussianNB()
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [25]:
```
def knn(X_train,y_train,X_test):

        # Fitting K-NN to the Training set
        from sklearn.neighbors import KNeighborsClassifier
        classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski',
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [26]:
```
def Decision(X_train,y_train,X_test):

        # Fitting K-NN to the Training set
        from sklearn.tree import DecisionTreeClassifier
        classifier = DecisionTreeClassifier(criterion = 'entropy', random_state
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [27]:
```
def random(X_train,y_train,X_test):

        # Fitting K-NN to the Training set
        from sklearn.ensemble import RandomForestClassifier
        classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entr
        classifier.fit(X_train, y_train)
        classifier,Accuracy,report,X_test,y_test,cm=cm_prediction(classifier,X_t
        return  classifier,Accuracy,report,X_test,y_test,cm
```

In [28]:
```
#for creating table and providing the score
def selectk_Classification(acclog,accsvml,accsvmnl,accknn,accnav,accdes,accrf):

    dataframe=pd.DataFrame(index=['ChiSquare'],columns=['Logistic','SVMl','SVMnl
#enumerate is for creating the index and providing the no in the table
    for number,idex in enumerate(dataframe.index):
        dataframe['Logistic'][idex]=acclog[number]
        dataframe['SVMl'][idex]=accsvml[number]
        dataframe['SVMnl'][idex]=accsvmnl[number]
        dataframe['KNN'][idex]=accknn[number]
        dataframe['Navie'][idex]=accnav[number]
        dataframe['Decision'][idex]=accdes[number]
        dataframe['Random'][idex]=accrf[number]
    return dataframe
```

In [30]:
```
dep.value_counts()
```

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

In [31]:
```
kbest=selectkbest(indep,dep,6)

acclog=[]
accsvml=[]
accsvmnl=[]
accknn=[]
accnav=[]
```

```
accdes=[]
accrf=[]
```

In [32]:
```
X_train, X_test, y_train, y_test=split_scalar(indep,dep)


classifier,Accuracy,report,X_test,y_test,cm=logistic(X_train,y_train,X_test)
acclog.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=svm_linear(X_train,y_train,X_test)
accsvml.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=svm_NL(X_train,y_train,X_test)
accsvmnl.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=knn(X_train,y_train,X_test)
accknn.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=Navie(X_train,y_train,X_test)
accnav.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=Decision(X_train,y_train,X_test)
accdes.append(Accuracy)

classifier,Accuracy,report,X_test,y_test,cm=random(X_train,y_train,X_test)
accrf.append(Accuracy)

result=selectk_Classification(acclog,accsvml,accsvmnl,accknn,accnav,accdes,accrf

result
```

```
C:\Users\DIKSHA\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:14
69: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DIKSHA\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:14
69: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DIKSHA\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:14
69: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[32]:

	Logistic	SVMl	SVMnl	KNN	Navie	Decision	Random
ChiSquare	0.59184	0.5952	0.64916	0.76756	0.60188	0.7706	0.7778

For this Dataset we got an average score for KNN,Decision and Random.

And we got a below average score for Logistic,Svml,SVMnland naive bayes

In [39]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(indep, dep, test_size=1/3, r
from sklearn.neighbors import KNeighborsClassifier
grid = KNeighborsClassifier()
grid.fit(X_train,y_train)
```

Out[39]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [41]:
```
#predict the xtest
y_pred = grid.predict(X_test)
```

In [42]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [45]:
```
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)  # Assuming X_train and y_train are your training data
y_pred = clf.predict(X_test)  # Assuming X_test is your test data
clf_report = classification_report(y_test, y_pred)
print(clf_report)
```

```
              precision    recall  f1-score   support

        Good       0.71      0.74      0.73      5927
        Poor       0.76      0.82      0.79      9718
    Standard       0.81      0.77      0.79     17689

    accuracy                           0.78     33334
   macro avg       0.76      0.78      0.77     33334
weighted avg       0.78      0.78      0.78     33334
```

In [50]:
```
print("Credit Score Prediction : ")
Annual_Income = float(input("Annual Income: "))
Salary = float(input("Monthly Inhand Salary: "))
No_Bank_Accounts = float(input("Number of Bank Accounts: "))
Intrest_Rate= float(input("Interest rate: "))
Loans= float(input("Number of Loans: "))
DelayedPayments= float(input("Number of delayed payments: "))
debt= float(input("Outstanding Debt: "))









Future_Prediction=grid.predict([[Annual_Income,Salary,No_Bank_Accounts,Intrest_R
print("Future_Prediction={}".format(Future_Prediction))
```

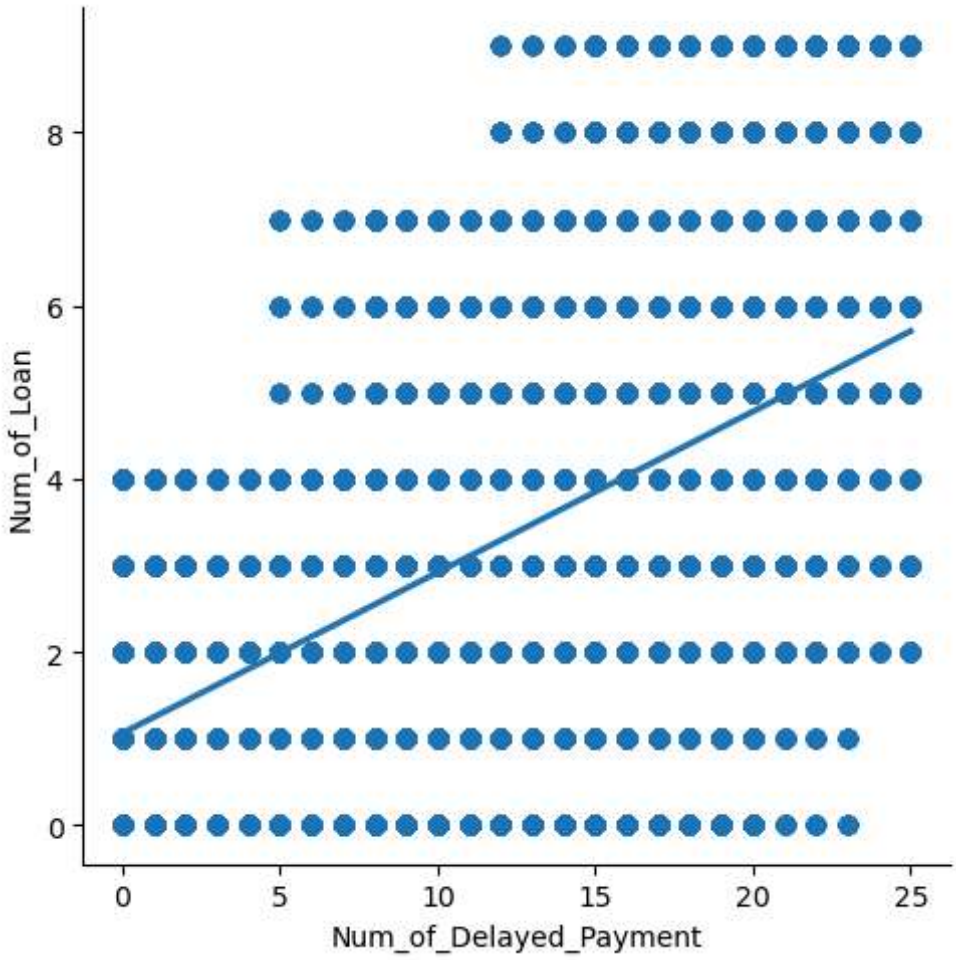
```
Credit Score Prediction :
Annual Income: 750000
Monthly Inhand Salary: 55000
Number of Bank Accounts: 4
Interest rate: 15
Number of Loans: 3
Number of delayed payments: 8
Outstanding Debt: 225000
Future_Prediction=['Poor']
```

```
C:\Users\DIKSHA\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X d
oes not have valid feature names, but KNeighborsClassifier was fitted with featur
e names
  warnings.warn(
```

In [17]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
dataset=pd.read_csv("Creditscore.csv")
sns.lmplot(x="Num_of_Delayed_Payment", y="Num_of_Loan", data=dataset)
plt.show
```

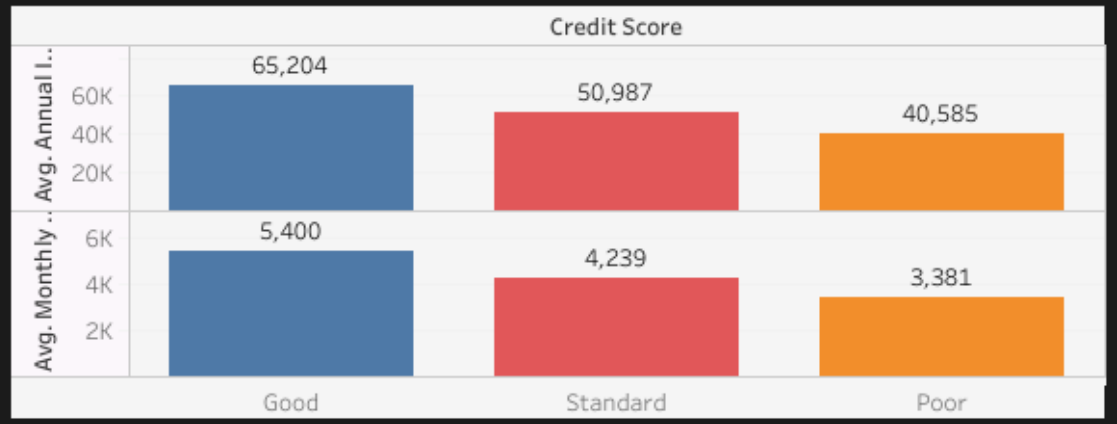
```
C:\Users\DIKSHA\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

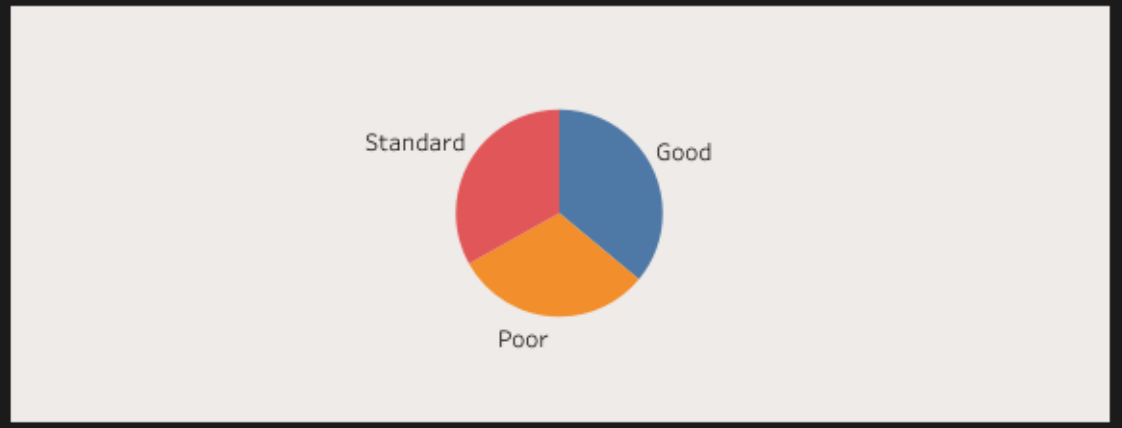


In [ ]:

Credit Analysis



Credit Score

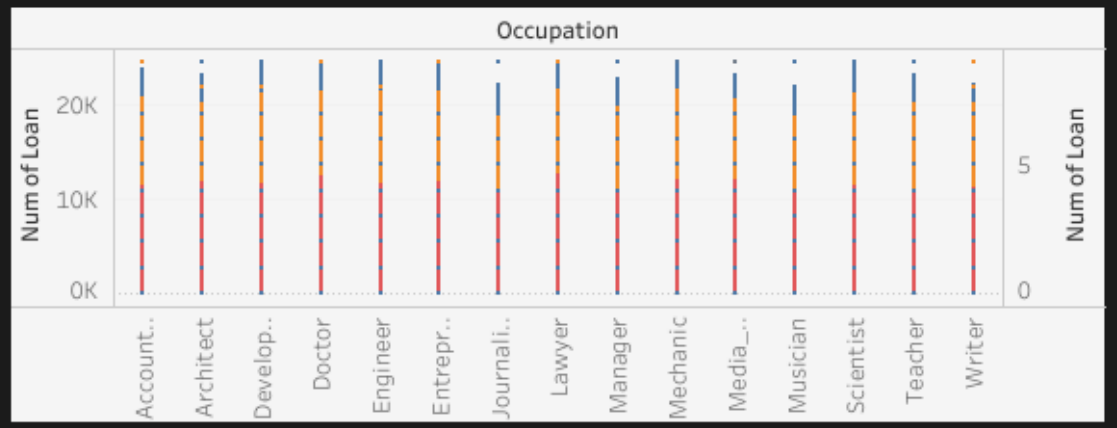


Measure Names
- Credit Card
- Bank Ac
- Loan

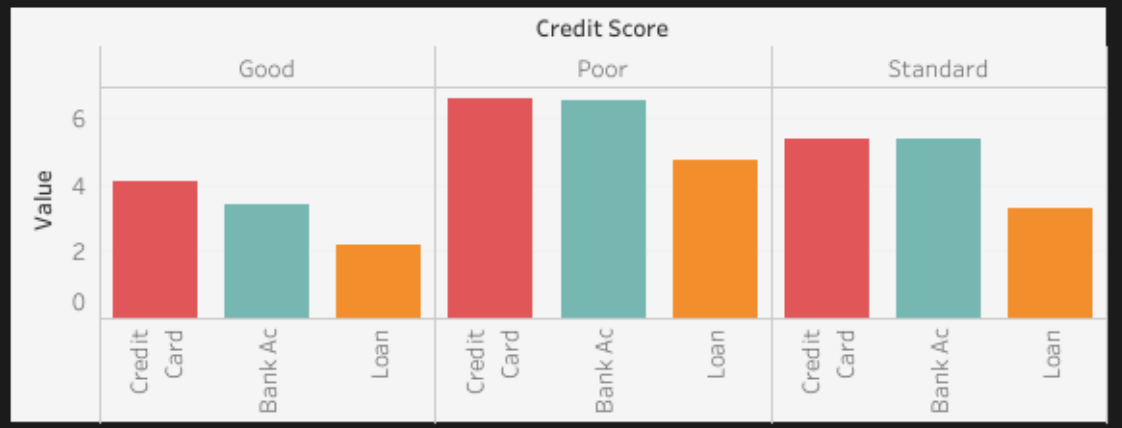
Avg. Monthly Inhand Sala..
3,380.560657177 to 5,3..
and Null values

Credit Score
- Good
- Poor
- Standard

Loan Analysis



Score VS Accounts



Credit score analysis using AI in Machine Learning

Credit score analysis using machine learning model in KNN algorithm" likely involves building a predictive model to assess creditworthiness based on historical data. Here's a brief overview of what such a project might entail:

Understanding the Problem: The project starts with a clear understanding of the problem statement. In this case, it involves predicting credit scores or assessing creditworthiness based on certain features or variables.

Data Collection: Gathering relevant data is crucial. This may include historical credit data such as credit scores, income levels, debt-to-income ratios, employment status, loan repayment history, etc. The data should be comprehensive and cover a representative sample of the population.

Data Preprocessing: Raw data often needs to be pre-processed before it can be used for modelling. This step involves handling missing values, dealing with outliers, encoding categorical variables, scaling numerical features, and possibly performing feature engineering to create new informative features.

Exploratory Data Analysis (EDA): EDA involves analysing and visualizing the data to gain insights. This step helps in understanding the relationships between different variables, identifying patterns, correlations, and potential predictive features.

Model Selection: Given the problem of credit score analysis, a supervised learning approach is suitable. Model selection involves choosing an appropriate algorithm. In this project, K-Nearest Neighbors (KNN) algorithm is chosen for its simplicity and effectiveness in classification tasks.

Model Training: The selected algorithm (KNN) is trained on the prepared dataset. During training, the model learns the patterns and relationships present in the data.

Model Evaluation: The trained model needs to be evaluated to assess its performance and generalization ability. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, and ROC-AUC.

Evaluation with KNN algorithm – KNN algorithm finds the nearby neighbours and provide the accuracy of the results, for this dataset we got accuracy report with 78% with support of 33334 datas value.

Hyperparameter Tuning: KNN algorithm has a hyperparameter 'k', which represents the number of neighbors to consider. Hyperparameter tuning involves selecting the optimal value of 'k' to achieve the best model performance.

To get the accuracy , I have used KNN Classifier to improve the score value.

KNN classifier Accuracy Score				
	precision	recall	f1-score	support
Good	0.71	0.74	0.73	5927
Poor	0.76	0.82	0.79	9718
Standard	0.81	0.77	0.79	17689
accuracy			0.78	17689
macro avg	0.76	0.78	0.77	33334
weighted avg	0.78	0.78	0.78	33334

KNN neighbor predicted the credit score and segregate the score value.

Model Deployment: Once the model is trained and evaluated satisfactorily, it can be deployed for real-world use. Deployment involves integrating the model into a software application or system where it can be used to make predictions on new data.