**Q1 Machine Learning Project Report: Credit Score Classification**

**By: Vatsal Sivaratri**

**9/18/2024**

**Table of Contents**

[**Part 1 – Statement/Project Goal 3**](#_heading=h.30j0zll)

[**Part 2 – Description of Dataset 3**](#_heading=h.1fob9te)

[**Part 3 – Methods and Materials**](#_heading=h.3znysh7) **3**

[**Part 4 – Results 14**](#_heading=h.3znysh7)

# 

# **1 Statement/Project Goal**

The goal of this project is to develop a predictive, robust model(s) for classifying individuals' credit scores. The motivation behind this project is the growing need for transparency and accuracy in evaluating trustworthiness of Credit Card users. A strong relationship between banks and their customers is necessary for a productive economy. The hope is to use past data on credit card holders to predict the likelihood of a good credit score in new customers, by considering a variety of attributes.

# **2 Description of Dataset**

The data was found on [Kaggle](https://www.kaggle.com/datasets/parisrohan/credit-score-classification/data) from a competition hosted by the website themselves. This dataset has conveniently been split into training and testing datasets. The training dataset contains 100,000 instances and 27 attributes (Dimension – 28). The attributes include both numerical data (e.g., monthly in-hand salary, outstanding debt) and categorical data (e.g., occupation, credit mix). Here’s a list of the key attributes:

**Age:** Age of individual

**Occupation:** Profession of Individual

**Annual\_Income:** Annual earnings of individual

**Monthly\_Inhand\_Salary:** Amount individual receives per month.

**Num\_Bank\_Accounts:** Number of bank accounts the individual holds

**Outstanding\_Debt:** Amount of unpaid debt

**Credit\_Utilization\_Ratio:** The ratio of the credit limit utilized

**Credit\_History\_Age:** The duration of the individual’s credit history

**Credit\_Mix:** Information on the mix of credit types (e.g., Good, Bad).

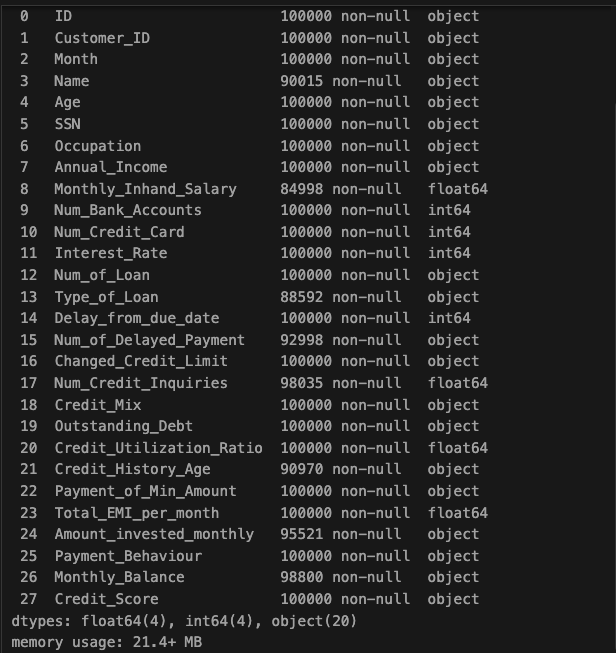
**Monthly\_Balance:** The remaining balance at the end of each month after all payments.

There are some interesting observations, including potential outliers in the age column (like negative values for age), repeated values in columns that are supposed to be unique (SSN), and missing values in some columns.

# **3 Methods and Materials**

**Data Preprocessing:**

When we first start with our dataset, we have a total of 27 features, 1 class variable, with 100,000 instances.

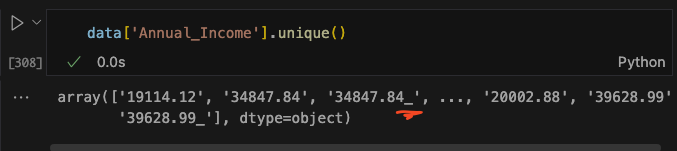


1. **Removing any Overtly Useless Columns:** Ex. A person’s SSN is useless in predicting their credit score without any other relevant information given by it (and even then it shouldn’t be needed as a feature itself)=> Go through all features and see which ones aren’t productive and need to be removed.

useless\_columns = ['ID', 'Customer\_ID', 'Name', 'SSN']

data.drop(useless\_columns, axis=1, inplace=True)

1. **Handling Underscore Issue:** In this dataset, there appears to be a continued problem of instances have an extra underscore, which prevents further analysis.



def remove\_trailing\_underscore(df, column\_name):

def clean\_value(value):

if isinstance(value, str) and value.endswith('\_'):

return value[:-1]

return value

df[column\_name] = df[column\_name].apply(clean\_value)

return df

for column in data.columns:

data = remove\_trailing\_underscore(data, column)

1. **Outlier Detection and Removal:** Certain variables like “Age” and “Monthly\_Balance” can have extreme values that are outliers and bad representations of other cars with similar specifications. These will need to be identified and removed, both manually and with IQR.

**Graph distributions of features to see any apparent outliers:**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**def plot\_feature\_distributions(data, plot\_type='hist'):**

**num\_columns = len(data.columns)**

**plt.figure(figsize=(16, num\_columns \* 4))**

**for i, column in enumerate(data.columns, 1):**

**plt.subplot(num\_columns, 1, i)**

**if plot\_type == 'hist':**

**sns.histplot(data[column], kde=True, bins=100)**

**plt.title(f'Distribution of {column}')**

**elif plot\_type == 'box':**

**sns.boxplot(x=data[column])**

**plt.title(f'Boxplot of {column}')**

**plt.tight\_layout()**

**plt.show()**

**plot\_feature\_distributions(data, plot\_type='hist')**

**Output:** [**https://drive.google.com/file/d/1r5UP0r1FlC\_KLzjFjZdry6x4WfDCn0qW/view?usp=sharing**](https://drive.google.com/file/d/1r5UP0r1FlC_KLzjFjZdry6x4WfDCn0qW/view?usp=sharing)

**Remove Filler Outliers:** Datasets like these often have people inputting obviously wrong data as placeholders. These methods take care of those issues.

def process\_age(data):

age = pd.to\_numeric(data['Age'], errors='coerce')

wrong\_ages = age[(age == -500) | (age>995) | (age<18)].index

data = data.drop(wrong\_ages)

data['Age'] = pd.to\_numeric(data['Age'], errors='coerce')

return data

data = process\_age(data)

def process\_ir(data):

ir = pd.to\_numeric(data['Interest\_Rate'], errors='coerce')

wrong\_irs = ir[(ir <0) | (ir>100)].index

data = data.drop(wrong\_irs)

data['Interest\_Rate'] = pd.to\_numeric(data['Interest\_Rate'], errors='coerce')

return data

data = process\_ir(data)

def convert\_credit\_history\_age\_to\_months(data):

def convert\_to\_months(value):

if pd.isna(value):

return None

parts = value.split('and')

years = int(parts[0].strip().split()[0])

months = int(parts[1].strip().split()[0])

return years \* 12 + months

data['Credit\_History\_Age'] = data['Credit\_History\_Age'].apply(convert\_to\_months)

data['Credit\_History\_Age'].fillna(data['Credit\_History\_Age'].mean(), inplace=True)

return data

data = convert\_credit\_history\_age\_to\_months(data)

def process\_delayed\_payment(data):

delayed\_payment = pd.to\_numeric(data['Num\_of\_Delayed\_Payment'], errors='coerce')

wrong\_irs = delayed\_payment[((delayed\_payment < 0))].index

data = data.drop(wrong\_irs)

data['Num\_of\_Delayed\_Payment'] = pd.to\_numeric(data['Num\_of\_Delayed\_Payment'], errors='coerce')

return data

data = process\_delayed\_payment(data)

def process\_amount\_invested\_monthly(data):

data['Amount\_invested\_monthly'] = data['Amount\_invested\_monthly'].replace('\_\_10000\_', pd.NA)

data['Amount\_invested\_monthly'] = pd.to\_numeric(data['Amount\_invested\_monthly'], errors='coerce')

return data

data = process\_amount\_invested\_monthly(data)

def process\_payment\_behavior(data):

data['Payment\_Behaviour'] = data['Payment\_Behaviour'].replace('!@9#%8', '0')

return data

data = process\_payment\_behavior(data)

For quantitative continuous features, also remove outliers with IQR:

**features\_to\_process = ['Age', 'Annual\_Income', 'Monthly\_Inhand\_Salary',**

**'Outstanding\_Debt', 'Credit\_Utilization\_Ratio',**

**'Total\_EMI\_per\_month', 'Amount\_invested\_monthly']**

**for feature in features\_to\_process:**

**data[feature] = pd.to\_numeric(data[feature], errors='coerce')**

**data[feature].fillna(data[feature].mean(), inplace=True)**

**Q1 = data[feature].quantile(0.25)**

**Q3 = data[feature].quantile(0.75)**

**IQR = Q3 - Q1**

**data = data[~((data[feature] < (Q1 - 1.5 \* IQR)) | (data[feature] > (Q3 + 1.5 \* IQR)))]**

1. **Fix Loan Feature Representations:** Loans are a very important feature in determining what kind of credit score a person has. However, the way loans are currently represented in the dataset combines all of them as one feature, making it hard to represent meaningfully to the model (Too many combinations of loan types). Thus, split this feature into 10 other features, 1 for each possible category:

def add\_loan\_type\_columns(data):

unique\_loan\_types = ['Auto Loan', 'Credit-Builder Loan', 'Debt Consolidation Loan', 'Home Equity Loan', 'Mortgage Loan',

'No Loan', 'Not Specified', 'Payday Loan', 'Personal Loan', 'Student Loan']

for loan\_type in unique\_loan\_types:

cleaned\_loan\_type = loan\_type.replace(' ', '\_').replace('-', '\_').lower()

data[cleaned\_loan\_type] = data['Type\_of\_Loan'].apply(lambda x: x.count(loan\_type) if isinstance(x, str) else 0)

data.drop(['Type\_of\_Loan'], axis=1, inplace=True)

return data

data = add\_loan\_type\_columns(data)

1. **Handle Missing Data:** As mentioned before, several columns contain missing values. How to fill them depends on the feature being dealt with. For example, the column “Monthly\_Inhand\_Salary” has missing values, and since this is a numerical, continuous feature, we may best fill them using the mean of the column. For “Num\_Credit\_Inquiries”, “Occupation” and other discrete quantitative or qualitative features, values will be filled with mode:

mean\_columns = ['Monthly\_Inhand\_Salary', 'Amount\_invested\_monthly', 'Monthly\_Balance', 'Changed\_Credit\_Limit']

mode\_columns = ['Num\_of\_Delayed\_Payment', 'Num\_Credit\_Inquiries']

categorical\_mode = ['Payment\_Behaviour']

for column in mean\_columns:

data[column] = pd.to\_numeric(data[column], errors='coerce')

data[column].fillna(data[column].mean(), inplace=True)

for column in mode\_columns:

data[column] = pd.to\_numeric(data[column], errors='coerce')

data[column].fillna(data[column].mode()[0], inplace=True)

1. **Encoding Categorical Variables:** Columns like “Occupation” and “Credit\_Mix” need to be encoded as numerical values for the model. This might involve one-hot encoding or label encoding.

toEncode = ['Month', 'Occupation','Credit\_Mix','Payment\_of\_Min\_Amount', 'Payment\_Behaviour', 'Credit\_Score']

label\_encoder = LabelEncoder()

for column in toEncode:

data[column] = label\_encoder.fit\_transform(data[column])

After preprocessing the data, we went from 100,000 instances and 27 features to 74652 instances and 32 features.

**Feature Selection: 4** Feature Selection Algorithms will be selected and applied onto the data through WEKA, and one data subset will be selected separately.

**Method 1:** CorrelationAttributeEval

0.18151 12 Changed\_Credit\_Limit

0.17729 11 Num\_of\_Delayed\_Payment

0.17391 18 Payment\_of\_Min\_Amount

0.16813 6 Num\_Bank\_Accounts

0.13492 14 Credit\_Mix

0.11204 8 Interest\_Rate

0.10648 7 Num\_Credit\_Card

0.08069 10 Delay\_from\_due\_date

0.05262 13 Num\_Credit\_Inquiries

0.04484 9 Num\_of\_Loan

0.02833 31 personal\_loan

0.0261 15 Outstanding\_Debt

0.02551 25 debt\_consolidation\_loan

0.02287 21 Payment\_Behaviour

0.02244 23 auto\_loan

0.02057 26 home\_equity\_loan

0.02003 24 credit\_builder\_loan

0.01703 27 mortgage\_loan

0.01683 30 payday\_loan

0.01486 29 not\_specified

0.00501 3 Occupation

0.00482 32 student\_loan

0.00318 1 Month

0 28 no\_loan

-0.00864 16 Credit\_Utilization\_Ratio

-0.01213 19 Total\_EMI\_per\_month

-0.02063 20 Amount\_invested\_monthly

-0.03455 5 Monthly\_Inhand\_Salary

-0.03914 4 Annual\_Income

-0.04084 22 Monthly\_Balance

-0.04399 2 Age

-0.08273 17 Credit\_History\_Age

**Cut-off: 0.10 => Selected Features:**

0.18151 12 Changed\_Credit\_Limit

0.17729 11 Num\_of\_Delayed\_Payment

0.17391 18 Payment\_of\_Min\_Amount

0.16813 6 Num\_Bank\_Accounts

0.13492 14 Credit\_Mix

0.11204 8 Interest\_Rate

0.10648 7 Num\_Credit\_Card

**Method 2:** ReliefAttributeEval (Ranker):

0.024477 2 Age

0.017906 7 Num\_Credit\_Card

0.017332 6 Num\_Bank\_Accounts

0.017219 8 Interest\_Rate

0.015441 15 Outstanding\_Debt

0.014112 19 Total\_EMI\_per\_month

0.014064 3 Occupation

0.0127 17 Credit\_History\_Age

0.011524 11 Num\_of\_Delayed\_Payment

0.01107 4 Annual\_Income

0.010864 10 Delay\_from\_due\_date

0.009616 13 Num\_Credit\_Inquiries

0.00737 12 Changed\_Credit\_Limit

0.006013 9 Num\_of\_Loan

0.005844 5 Monthly\_Inhand\_Salary

0.005668 32 student\_loan

0.005657 30 payday\_loan

0.005379 29 not\_specified

0.005245 27 mortgage\_loan

0.005222 24 credit\_builder\_loan

0.00521 23 auto\_loan

0.005032 31 personal\_loan

0.004947 25 debt\_consolidation\_loan

0.0048 26 home\_equity\_loan

0 28 no\_loan

-0.000387 22 Monthly\_Balance

-0.009254 20 Amount\_invested\_monthly

-0.018253 16 Credit\_Utilization\_Ratio

-0.021545 18 Payment\_of\_Min\_Amount

-0.023212 14 Credit\_Mix

-0.032999 21 Payment\_Behaviour

-0.060802 1 Month

**Cut-off: 0.02 => Selected Features:**

0.024477 2 Age

-0.021545 18 Payment\_of\_Min\_Amount

-0.023212 14 Credit\_Mix

-0.032999 21 Payment\_Behaviour

-0.060802 1 Month

**Method 3:** ClassifierAttributeEval (Ranker)

0.1879941 4 Annual\_Income

0.1812442 15 Outstanding\_Debt

0.1583674 19 Total\_EMI\_per\_month

0.1448428 5 Monthly\_Inhand\_Salary

0.0667689 14 Credit\_Mix

0.0327237 8 Interest\_Rate

0.0273498 7 Num\_Credit\_Card

0.0242182 18 Payment\_of\_Min\_Amount

0.0229147 11 Num\_of\_Delayed\_Payment

0.0222949 6 Num\_Bank\_Accounts

0.0188205 10 Delay\_from\_due\_date

0.0187684 12 Changed\_Credit\_Limit

0.0048473 13 Num\_Credit\_Inquiries

0.0016956 2 Age

0.001023 9 Num\_of\_Loan

0.0004473 1 Month

0.0003747 31 personal\_loan

0.000297 25 debt\_consolidation\_loan

0.0002325 23 auto\_loan

0.0002311 17 Credit\_History\_Age

0.000171 21 Payment\_Behaviour

0.0001605 29 not\_specified

0.0001526 3 Occupation

0.0001436 27 mortgage\_loan

0.000138 26 home\_equity\_loan

0.0001307 24 credit\_builder\_loan

0.0000864 30 payday\_loan

0.0000121 32 student\_loan

0 28 no\_loan

-0.1479401 20 Amount\_invested\_monthly

-0.1575615 22 Monthly\_Balance

-0.1602462 16 Credit\_Utilization\_Ratio

**Cutoff: 0.10 => Selected Features:**

0.1879941 4 Annual\_Income

0.1812442 15 Outstanding\_Debt

0.1583674 19 Total\_EMI\_per\_month

0.1448428 5 Monthly\_Inhand\_Salary

-0.1479401 20 Amount\_invested\_monthly

-0.1575615 22 Monthly\_Balance

-0.1602462 16 Credit\_Utilization\_Ratio

**Method 4:** CfsSubsetEval (GreedyStepwise)

**Algorithm’s Selected Features:**

Num\_Bank\_Accounts

Num\_of\_Delayed\_Payment

Changed\_Credit\_Limit

Credit\_Mix

Payment\_of\_Min\_Amount

**Method 5** (Selecting Subset Manually): 'Annual\_Income','Monthly\_Inhand\_Salary','Num\_of\_Loan',

'Outstanding\_Debt','Credit\_Utilization\_Ratio','Credit\_History\_Age',

'Num\_Credit\_Card','Num\_of\_Delayed\_Payment','Interest\_Rate','Total\_EMI\_per\_month'

I selected these features from research on the biggest influencers of Credit Score combined with my own intuition.

**Models:** Using Decision Tree, Random Forest, SVM, Logistic Regression, and Naive Bayes.

**For Metrics:** Accuracy to evaluate overall performance, ROC-AUCs to see how well the model tells the difference between classes, and TP and FP rates.

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**models = {**

**"Decision Tree": DecisionTreeClassifier(),**

**"Random Forest": RandomForestClassifier(),**

**"Logistic Regression": LogisticRegression(max\_iter=1000),**

**"Naive Bayes": GaussianNB(),**

**"SVM": SVC(probability=True)**

**}**

**results = {}**

**for model\_name, model in models.items():**

**model.fit(X\_train, y\_train)**

**y\_pred = model.predict(X\_test)**

**y\_pred\_proba = model.predict\_proba(X\_test) if hasattr(model, 'predict\_proba') else None**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**cm = confusion\_matrix(y\_test, y\_pred)**

**tpr = {}**

**fpr = {}**

**for i in range(cm.shape[0]):**

**tp = cm[i, i]**

**fn = cm[i, :].sum() - tp**

**fp = cm[:, i].sum() - tp**

**tn = cm.sum() - (tp + fn + fp)**

**tpr[i] = tp / (tp + fn) if (tp + fn) != 0 else 0**

**fpr[i] = fp / (fp + tn) if (fp + tn) != 0 else 0**

**roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba, multi\_class='ovr') if y\_pred\_proba is not None else 'N/A'**

**results[model\_name] = {**

**'Accuracy': accuracy,**

**'Confusion Matrix': cm,**

**'True Positive Rate': tpr,**

**'False Positive Rate': fpr,**

**'ROC-AUC': roc\_auc**

**}**

**for model\_name, metrics in results.items():**

**print(f"Model: {model\_name}")**

**print(f"Accuracy: {metrics['Accuracy']}")**

**print(f"Confusion Matrix: \n{metrics['Confusion Matrix']}")**

**print(f"True Positive Rate (TPR): {metrics['True Positive Rate']}")**

**print(f"False Positive Rate (FPR): {metrics['False Positive Rate']}")**

**print(f"ROC-AUC: {metrics['ROC-AUC']}\n")**

**best\_model = max(results, key=lambda x: results[x]['Accuracy'])**

**print(f"The best model is: {best\_model}")**

# **4 Results**

**CorrelationAttributeEval**

| **Model** | **Accuracy** | **ROC-AUC** | **TPR\_Class\_0** | **FPR\_Class\_0** | **TPR\_Class\_1** | **FPR\_Class\_1** | **TPR\_Class\_2** | **FPR\_Class\_2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.688 | 0.755 | 0.624 | 0.09 | 0.682 | 0.149 | 0.71 | 0.285 |
| Random Forest | 0.738 | 0.871 | 0.644 | 0.07 | 0.699 | 0.103 | 0.787 | 0.284 |
| Logistic Regression | 0.637 | 0.778 | 0.365 | 0.06 | 0.474 | 0.096 | 0.806 | 0.533 |
| Naive Bayes | 0.629 | 0.769 | 0.754 | 0.186 | 0.612 | 0.151 | 0.6 | 0.236 |
| SVM | 0.659 | 0.793 | 0.358 | 0.057 | 0.498 | 0.073 | 0.836 | 0.527 |

# 

# **ReliefAttributeEval**

| **Model** | **Accuracy** | **ROC-AUC** | **TPR\_Class\_0** | **FPR\_Class\_0** | **TPR\_Class\_1** | **FPR\_Class\_1** | **TPR\_Class\_2** | **FPR\_Class\_2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.539 | 0.677 | 0.54 | 0.148 | 0.438 | 0.216 | 0.593 | 0.402 |
| Random Forest | 0.554 | 0.709 | 0.478 | 0.118 | 0.39 | 0.181 | 0.664 | 0.477 |
| Logistic Regression | 0.528 | 0.674 | 0.068 | 0.046 | 0.232 | 0.11 | 0.825 | 0.778 |
| Naive Bayes | 0.556 | 0.714 | 0.135 | 0.028 | 0.273 | 0.112 | 0.834 | 0.747 |
| SVM | 0.544 | 0.718 | 0 | 0 | 0 | 0 | 1 | 1 |

**ClassifierAttributeEval**

| **Model** | **Accuracy** | **ROC-AUC** | **TPR\_Class\_0** | **FPR\_Class\_0** | **TPR\_Class\_1** | **FPR\_Class\_1** | **TPR\_Class\_2** | **FPR\_Class\_2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.697 | 0.751 | 0.615 | 0.083 | 0.687 | 0.126 | 0.726 | 0.313 |
| Random Forest | 0.719 | 0.861 | 0.378 | 0.033 | 0.721 | 0.104 | 0.821 | 0.391 |
| Logistic Regression | 0.551 | 0.668 | 0.004 | 0.001 | 0.284 | 0.111 | 0.859 | 0.807 |
| Naive Bayes | 0.544 | 0.66 | 0.233 | 0.133 | 0.588 | 0.18 | 0.614 | 0.474 |
| SVM | 0.544 | 0.69 | 0.006 | 0.002 | 0.231 | 0.098 | 0.874 | 0.841 |

**CfsSubsetEval**

| **Model** | **Accuracy** | **ROC-AUC** | **TPR\_Class\_0** | **FPR\_Class\_0** | **TPR\_Class\_1** | **FPR\_Class\_1** | **TPR\_Class\_2** | **FPR\_Class\_2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.638 | 0.716 | 0.57 | 0.101 | 0.63 | 0.184 | 0.662 | 0.32 |
| Random Forest | 0.635 | 0.794 | 0.521 | 0.089 | 0.577 | 0.17 | 0.701 | 0.37 |
| Logistic Regression | 0.588 | 0.724 | 0.276 | 0.066 | 0.36 | 0.099 | 0.805 | 0.625 |
| Naive Bayes | 0.59 | 0.731 | 0.662 | 0.164 | 0.474 | 0.151 | 0.63 | 0.361 |
| SVM | 0.603 | 0.741 | 0.386 | 0.094 | 0.304 | 0.056 | 0.829 | 0.608 |

# **My Feature Selection:**

| **Model** | **Accuracy** | **ROC-AUC** | **TPR\_Class\_0** | **FPR\_Class\_0** | **TPR\_Class\_1** | **FPR\_Class\_1** | **TPR\_Class\_2** | **FPR\_Class\_2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 0.727 | 0.774 | 0.644 | 0.074 | 0.714 | 0.111 | 0.759 | 0.287 |
| Random Forest | 0.793 | 0.913 | 0.706 | 0.045 | 0.801 | 0.091 | 0.815 | 0.228 |
| Logistic Regression | 0.569 | 0.731 | 0.16 | 0.04 | 0.353 | 0.116 | 0.809 | 0.689 |
| Naive Bayes | 0.587 | 0.751 | 0.757 | 0.256 | 0.69 | 0.167 | 0.481 | 0.173 |
| SVM | 0.543 | 0.69 | 0.006 | 0.002 | 0.23 | 0.099 | 0.874 | 0.842 |

# 

# 