CS-UY 4563 Machine Learning

Topic: Predicting Credit Card Approvals

Professor: Linda N. Sellie

Student: Giancarlo Enriquez

Email: enriquez@nyu.edu

1. Introduction

Have you ever applied for an apple credit card and after filling out the application you get denied immediately? How do banks quickly come up with this decision? Well, in today's society commercial banks receive thousands of applications for credit cards each month. Whether a person gets approved or declined really depends on the individual's financial situation, this includes, high loan balances, low income levels, or two many inquiries on an individual's credit report. While banks can manually analyze these applications individually, the drawbacks include mundane, error-prone, and time consuming. In this machine learning project, we will build an automatic credit card approval predictor using classification techniques from machine learning, just like the real banks do.

II. Dataset

Our data used in this project is from the UCI Machine Learning Repository called "Credit Card Approval dataset". We have to note that in our data set all attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. However, using "Analysis of Credit Approval Data" we can get the sense of an idea of what each column represents.

The table below describes the attributes used in this project.

Table 1: Attribute Information

A1: Gender	b,a
A2: Age	continuous
A3: Debt	continuous
A4: Married	u,y,l,t
A5: Bank Customer	g,p,gg
A6: Education Level	c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7: Ethnicity	v, h, bb, j, n, z, dd, ff, o.
A8: YearsEmployed	continuous
A9: PriorDefault	t, f.
A10: Employed	t, f.
A11: CreditScore	continuous.
A12: DriversLicense	t, f.
A13: Citizen	g, p, s.
A14: ZipCode	continuous.
A15: Income	continuous.
A16: ApprovalStatus	+,- (class attribute)

A. Now What? Explore the Dataset

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

Figure 1: The Dataset

As we look through our dataset, the dataset has a mixture of numerical and non-numerical features:

Gender	object
Age	object
Debt	float64
Married	object
BankCustomer	object
EducationLevel	object
Ethnicity	object
YearsEmployed	float64
PriorDefault	object
Employed	object
CreditScore	int64
DriversLicense	object
Citizen	object
ZipCode	object
Income	int64
ApprovalStatus	object

What is going on with our data? Is there any significance here? As we explore the data we know that the column ApprovalStatus has more (-) than (+) which is a good thing to see since we want to predict our model whether a person would be approved or declined. (The more data for declined the better estimate on our machine learning model)

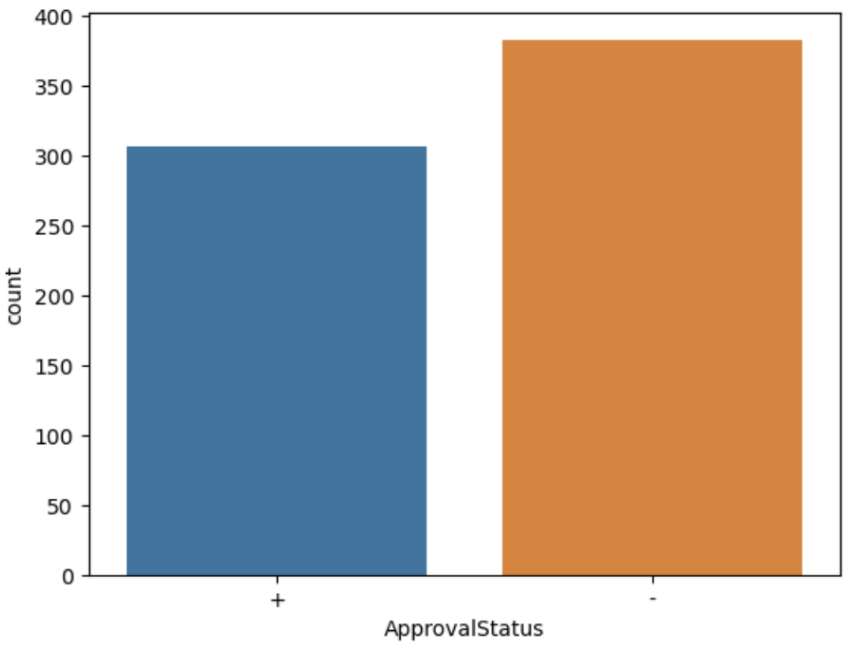


Figure 2: Counts of ApprovalStatus

If we look deeper, we will see that 55.5% got declined and 44.5% got approved. This can be modeled as a pie chart.

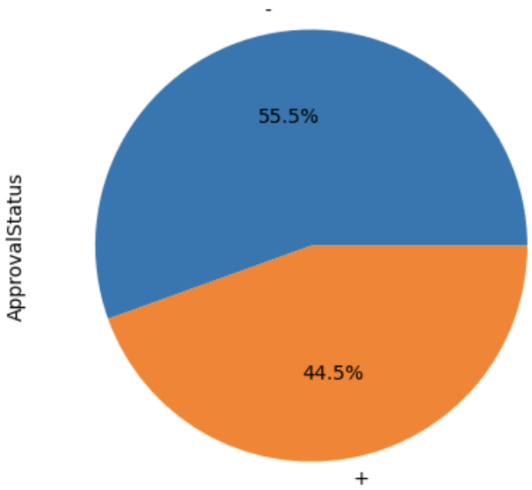


Figure 3: Pie Chart of ApprovalStatus

Since we have values for columns: Debt, YearsEmployed, CreditScore, and Income; we can use the .describe() function to generate descriptive statistics of the columns. We notice that the dataset contains values from several ranges. Some features have a value range of 0-28, some have a range of 0-67, and some have a range of 0-100000.

Figure 1 shows the Credit Cards Description

	Debt	YearsEmployed	CreditScore	Income
count	690.000000	690.000000	690.00000	690.000000
mean	4.758725	2.223406	2.40000	1017.385507
std	4.978163	3.346513	4.86294	5210.102598
min	0.000000	0.000000	0.00000	0.000000
25%	1.000000	0.165000	0.00000	0.000000
50%	2.750000	1.000000	0.00000	5.000000
75%	7.207500	2.625000	3.00000	395.500000
max	28.000000	28.500000	67.00000	100000.000000

Figure 4: Descriptive Statistics

Now that we have learned about our data. We need to check if there are any missing values in our dataset. To do this we use the .value_counts() method.

B. Inspect our Dataset. Any missing values?

```
b    468
a    210
?     12
Name: Gender, dtype: int64
```

```
?        12
22.67     9
20.42     7
18.83     6
24.50     6
         ..
48.25     1
28.33     1
18.75     1
18.50     1
36.42     1
Name: Age, Length: 350, dtype: int64
```

```
c     137
q      78
w      64
i      59
aa     54
ff     53
k      51
cc     41
m      38
x      38
d      30
e      25
j      10
?       9
r       3
Name: EducationLevel, dtype: int64
```

```
0.000    70
0.250    35
0.040    33
1.000    31
0.125    30
         ..
4.165     1
9.000     1
1.960     1
5.125     1
8.290     1
Name: YearsEmployed, Length: 132, dtype: int64
```

```
f    374
t    316
Name: DriversLicense, dtype: int64
```

```
g    625
s     57
p      8
Name: Citizen, dtype: int64
```

```
1.500     21
0.000     19
3.000     19
2.500     19
0.750     16
          ..
0.085      1
12.250     1
11.045     1
11.125     1
3.375      1
Name: Debt, Length: 215, dtype: int64
```

```
u    519
y    163
?      6
l      2
Name: Married, dtype: int64
```

```
g     519
p     163
?       6
gg      2
Name: BankCustomer, dtype: int64
```

```
v     399
h     138
bb     59
ff     57
?       9
j       8
z       8
dd      6
n       4
o       2
Name: Ethnicity, dtype: int64
```

```
00000    132
00200     35
00120     35
00160     34
00100     30
          ...
00021      1
00393      1
00395      1
00093      1
00256      1
Name: ZipCode, Length: 171, dtype: int64
```

```
0       295
1        29
500      10
1000     10
2         9
        ...
1704      1
857       1
6700      1
2503      1
750       1
Name: Income, Length: 240, dtype: int64
```

```
0     395
1      71
2      45
3      28
6      23
11     19
5      18
7      16
4      15
9      10
8      10
10      8
14      8
12      8
15      4
16      3
20      2
17      2
23      1
40      1
67      1
13      1
19      1
Name: CreditScore, dtype: int64
```

```
t    361
f    329
Name: PriorDefault, dtype: int64
```

```
f    395
t    295
Name: Employed, dtype: int64
```

```
-    383
+    307
Name: ApprovalStatus, dtype: int64
```

Figure 5: What's in our dataset? Missing values?

As we can clearly see, some columns can affect the performance of our machine learning model if we do not change it. We see that the dataset does indeed have missing values, these are labeled as '?'. Below you will find the dataset that has a '?' in the gender column, for example. We see that on row 673, Gender has a '?'.

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
673	?	29.50	2.000	y	p	e	h	2.000	f	f	0	f	g	00256	17	-
674	a	37.33	2.500	u	g	i	h	0.210	f	f	0	f	g	00260	246	-
675	a	41.58	1.040	u	g	aa	v	0.665	f	f	0	f	g	00240	237	-
676	a	30.58	10.665	u	g	q	h	0.085	f	t	12	t	g	00129	3	-
677	b	19.42	7.250	u	g	m	v	0.040	f	t	1	f	g	00100	1	-
678	a	17.92	10.210	u	g	ff	ff	0.000	f	f	0	f	g	00000	50	-
679	a	20.08	1.250	u	g	c	v	0.000	f	f	0	f	g	00000	0	-
680	b	19.50	0.290	u	g	k	v	0.290	f	f	0	f	g	00280	364	-
681	b	27.83	1.000	y	p	d	h	3.000	f	f	0	f	g	00176	537	-
682	b	17.08	3.290	u	g	i	v	0.335	f	f	0	t	g	00140	2	-
683	b	36.42	0.750	y	p	d	v	0.585	f	f	0	f	g	00240	3	-
684	b	40.58	3.290	u	g	m	v	3.500	f	f	0	t	s	00400	0	-
685	b	21.08	10.085	y	p	e	h	1.250	f	f	0	f	g	00260	0	-
686	a	22.67	0.750	u	g	c	v	2.000	f	t	2	t	g	00200	394	-
687	a	25.25	13.500	y	p	ff	ff	2.000	f	t	1	t	g	00200	1	-
688	b	17.92	0.205	u	g	aa	v	0.040	f	f	0	f	g	00280	750	-
689	b	35.00	3.375	u	g	c	h	8.290	f	f	0	t	g	00000	0	-

Figure 6: Where are the missing values located?

C. How to work with Missing Values?

How do we fix this problem? In order to fix this problem, we temporarily replace these missing value question marks with NaN, as shown in Figure 7.

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
673	NaN	29.50	2.000	y	p	e	h	2.000	f	f	0	f	g	00256	17	-
674	a	37.33	2.500	u	g	i	h	0.210	f	f	0	f	g	00260	246	-
675	a	41.58	1.040	u	g	aa	v	0.665	f	f	0	f	g	00240	237	-
676	a	30.58	10.665	u	g	q	h	0.085	f	t	12	t	g	00129	3	-
677	b	19.42	7.250	u	g	m	v	0.040	f	t	1	f	g	00100	1	-
678	a	17.92	10.210	u	g	ff	ff	0.000	f	f	0	f	g	00000	50	-
679	a	20.08	1.250	u	g	c	v	0.000	f	f	0	f	g	00000	0	-
680	b	19.50	0.290	u	g	k	v	0.290	f	f	0	f	g	00280	364	-
681	b	27.83	1.000	y	p	d	h	3.000	f	f	0	f	g	00176	537	-
682	b	17.08	3.290	u	g	i	v	0.335	f	f	0	t	g	00140	2	-
683	b	36.42	0.750	y	p	d	v	0.585	f	f	0	f	g	00240	3	-
684	b	40.58	3.290	u	g	m	v	3.500	f	f	0	t	s	00400	0	-
685	b	21.08	10.085	y	p	e	h	1.250	f	f	0	f	g	00260	0	-
686	a	22.67	0.750	u	g	c	v	2.000	f	t	2	t	g	00200	394	-
687	a	25.25	13.500	y	p	ff	ff	2.000	f	t	1	t	g	00200	1	-
688	b	17.92	0.205	u	g	aa	v	0.040	f	f	0	f	g	00280	750	-
689	b	35.00	3.375	u	g	c	h	8.290	f	f	0	t	g	00000	0	-

Figure 7: Replacing '?' with NaN

We use the np.nan method to replace '?' since it acts like a placeholder to represent the missing data in a data frame which is a floating-point number. But why do we care about missing values in our dataset? Can we just get rid/ignore them? The answer is no, of course not. By ignoring the missing values, it can affect the performance of our machine learning model heavily. Three problems may arise if we ignore, (1) it reduces the statistical power of our model, (2) it may miss out on information about the dataset that may be useful for its training and (3) it can produce biased estimates leading to invalid conclusions. Models like LDA (Linear Discriminant Analysis) cannot handle this problem. However, algorithms like K-nearest and Naive Bayes (beyond the scope of this project) support data with missing values. So, how do we avoid this problem?

To avoid building a biased machine learning model (which will lead to incorrect results if the missing values are not handled properly), we will impute the missing values with a strategy called mean imputation on the numeric columns. What is so great about mean imputation? The mean preserves the mean of the observed data. So, if the data is missing completely at random, the estimate of the mean remains unbiased. Also, by imputing the mean, we are able to keep the sample size up to the full sample size.

What do we do for missing values for non-numeric columns? We are going to impute the missing values with the most frequent values as present in the respective columns. This can be seen in Figure 8.

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
673	b	29.50	2.000	y	p	e	h	2.000	f	f	0	f	g	00256	17	-
674	a	37.33	2.500	u	g	i	h	0.210	f	f	0	f	g	00260	246	-
675	a	41.58	1.040	u	g	aa	v	0.665	f	f	0	f	g	00240	237	-
676	a	30.58	10.665	u	g	q	h	0.085	f	t	12	t	g	00129	3	-
677	b	19.42	7.250	u	g	m	v	0.040	f	t	1	f	g	00100	1	-
678	a	17.92	10.210	u	g	ff	ff	0.000	f	f	0	f	g	00000	50	-
679	a	20.08	1.250	u	g	c	v	0.000	f	f	0	f	g	00000	0	-
680	b	19.50	0.290	u	g	k	v	0.290	f	f	0	f	g	00280	364	-
681	b	27.83	1.000	y	p	d	h	3.000	f	f	0	f	g	00176	537	-
682	b	17.08	3.290	u	g	i	v	0.335	f	f	0	t	g	00140	2	-
683	b	36.42	0.750	y	p	d	v	0.585	f	f	0	f	g	00240	3	-
684	b	40.58	3.290	u	g	m	v	3.500	f	f	0	t	s	00400	0	-
685	b	21.08	10.085	y	p	e	h	1.250	f	f	0	f	g	00260	0	-
686	a	22.67	0.750	u	g	c	v	2.000	f	t	2	t	g	00200	394	-
687	a	25.25	13.500	y	p	ff	ff	2.000	f	t	1	t	g	00200	1	-
688	b	17.92	0.205	u	g	aa	v	0.040	f	f	0	f	g	00280	750	-
689	b	35.00	3.375	u	g	c	h	8.290	f	f	0	t	g	00000	0	-

Figure 8: Fixing Missing Values in Our Dataset

D. Preprocessing the Data

Now that we have fixed our missing values, what else can we do? Well in order to model this data, we need to preprocess the data in three steps: (1) convert the non-numeric data into numeric, (2) Split the data into train and test sets, and (3) scale the features values using Standard Scaler. Let's first convert all the non-numerical values into numeric ones using a strategy called LabelEncoder from scikit learn. Why do we use Label Encoding? Well, label encoding is used for categorical variables. Label encoding is easy to implement and interpret, it is visually user friendly and it works best with a smaller number of unique categorical values. We use this label encoding since many machine learning models require the data to be strictly numeric format, which produces results faster when computing.

Using LabelEncoder, we get our new improved data (see Figure 9)

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
0	1	156	0.000	2	1	13	8	1.25	1	1	1	0	0	68	0	0
1	0	328	4.460	2	1	11	4	3.04	1	1	6	0	0	11	560	0
2	0	89	0.500	2	1	11	4	1.50	1	0	0	0	0	96	824	0
3	1	125	1.540	2	1	13	8	3.75	1	1	5	1	0	31	3	0
4	1	43	5.625	2	1	13	8	1.71	1	0	0	0	2	37	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	1	52	10.085	3	3	5	4	1.25	0	0	0	0	0	90	0	1
686	0	71	0.750	2	1	2	8	2.00	0	1	2	1	0	67	394	1
687	0	97	13.500	3	3	6	3	2.00	0	1	1	1	0	67	1	1
688	1	20	0.205	2	1	0	8	0.04	0	0	0	0	0	96	750	1
689	1	197	3.375	2	1	2	4	8.29	0	0	0	1	0	0	0	1

690 rows × 16 columns

Figure 9: Our data with Label Encoding

Let's now analyze our data with the .corr() function. We want to find which columns gives us a correlation towards ApprovalStatus (our endgame). As we see in Figure 10, there seems to be no correlation between DriversLicense vs ApprovalStatus and Zipcode vs Approval Status. So we can drop these features.

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	ApprovalStatus
Gender	1.000000	0.062296	-0.041746	0.068365	0.069627	-0.012486	0.044686	0.086544	-0.026047	-0.077784	-0.024630	0.051674	0.085488	0.097600	-0.002063	0.028934
Age	0.062296	1.000000	0.135058	-0.068214	-0.085631	0.032039	-0.190155	0.386076	0.197493	0.047300	0.160599	0.079829	0.001284	-0.001211	0.016829	-0.133304
Debt	-0.041746	0.135058	1.000000	-0.047608	-0.068773	0.023428	-0.016451	0.298902	0.244317	0.174846	0.271207	-0.013023	-0.122233	-0.262772	0.123121	-0.206294
Married	0.068365	-0.068214	-0.047608	1.000000	0.942463	0.001738	0.063158	-0.048423	-0.078851	-0.114926	-0.077948	0.029057	-0.094585	-0.091238	-0.101102	0.191431
BankCustomer	0.069627	-0.085631	-0.068773	0.942463	1.000000	-0.041508	-0.003989	-0.065497	-0.129863	-0.162464	-0.106457	0.015342	-0.036095	-0.018734	-0.022904	0.187520
EducationLevel	-0.012486	0.032039	0.023428	0.001738	-0.041508	1.000000	0.038218	0.041492	0.107793	0.132133	0.012271	0.075946	-0.010663	0.070429	0.007381	-0.130026
Ethnicity	0.044686	-0.190155	-0.016451	0.063158	-0.003989	0.038218	1.000000	-0.074281	-0.002135	0.032440	-0.015461	0.020007	-0.016338	-0.064167	-0.009131	-0.000877
YearsEmployed	0.086544	0.386076	0.298902	-0.048423	-0.065497	0.041492	-0.074281	1.000000	0.345689	0.222982	0.322330	0.138139	-0.020528	-0.106919	0.051345	-0.322475
PriorDefault	-0.026047	0.197493	0.244317	-0.078851	-0.129863	0.107793	-0.002135	0.345689	1.000000	0.432032	0.379532	0.091276	-0.113623	-0.096796	0.090012	-0.720407
Employed	-0.077784	0.047300	0.174846	-0.114926	-0.162464	0.132133	0.032440	0.222982	0.432032	1.000000	0.571498	0.017043	-0.242568	-0.091529	0.077652	-0.458301
CreditScore	-0.024630	0.160599	0.271207	-0.077948	-0.106457	0.012271	-0.015461	0.322330	0.379532	0.571498	1.000000	0.006944	-0.139527	-0.139028	0.063692	-0.406410
DriversLicense	0.051674	0.079829	-0.013023	0.029057	0.015342	0.075946	0.020007	0.138139	0.091276	0.017043	0.006944	1.000000	0.005883	0.137117	0.019201	-0.031625
Citizen	0.085488	0.001284	-0.122233	-0.094585	-0.036095	-0.010663	-0.016338	-0.020528	-0.113623	-0.242568	-0.139527	0.005883	1.000000	0.100442	-0.012017	0.100867
ZipCode	0.097600	-0.001211	-0.262772	-0.091238	-0.018734	0.070429	-0.064167	-0.106919	-0.096796	-0.091529	-0.139028	0.137117	0.100442	1.000000	0.088546	0.094851
Income	-0.002063	0.016829	0.123121	-0.101102	-0.022904	0.007381	-0.009131	0.051345	0.090012	0.077652	0.063692	0.019201	-0.012017	0.088546	1.000000	-0.175657
ApprovalStatus	0.028934	-0.133304	-0.206294	0.191431	0.187520	-0.130026	-0.000877	-0.322475	-0.720407	-0.458301	-0.406410	-0.031625	0.100867	0.094851	-0.175657	1.000000

Figure 10: Correlation of the columns

Dropping the columns: DriversLicense and Zipcode we get that the dataset we modeling is (shown in Figure 11):

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity
0	1	156	0.000	2	1	13	8
1	0	328	4.460	2	1	11	4
2	0	89	0.500	2	1	11	4
3	1	125	1.540	2	1	13	8
4	1	43	5.625	2	1	13	8
..	...	...	...	...	...	...	...
685	1	52	10.085	3	3	5	4
686	0	71	0.750	2	1	2	8
687	0	97	13.500	3	3	6	3
688	1	20	0.205	2	1	0	8
689	1	197	3.375	2	1	2	4

	YearsEmployed	PriorDefault	Employed	CreditScore	Citizen	Income
0	1.25	1	1	1	0	0
1	3.04	1	1	6	0	560
2	1.50	1	0	0	0	824
3	3.75	1	1	5	0	3
4	1.71	1	0	0	2	0
..	...	...	...	...	...	...
685	1.25	0	0	0	0	0
686	2.00	0	1	2	0	394
687	2.00	0	1	1	0	1
688	0.04	0	0	0	0	750
689	8.29	0	0	0	0	0

	ApprovalStatus
0	0
1	0
2	0
3	0
4	0
..	...
685	1
686	1
687	1
688	1
689	1

Figure 11

Now we can split our dataset into train and test sets with a test size of 0.33 and a random state of 42. Let's see the density of our training set and test set.

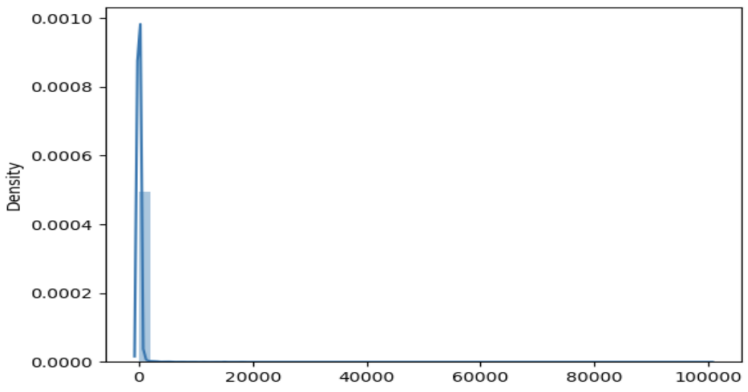


Figure 12: Density of our training set

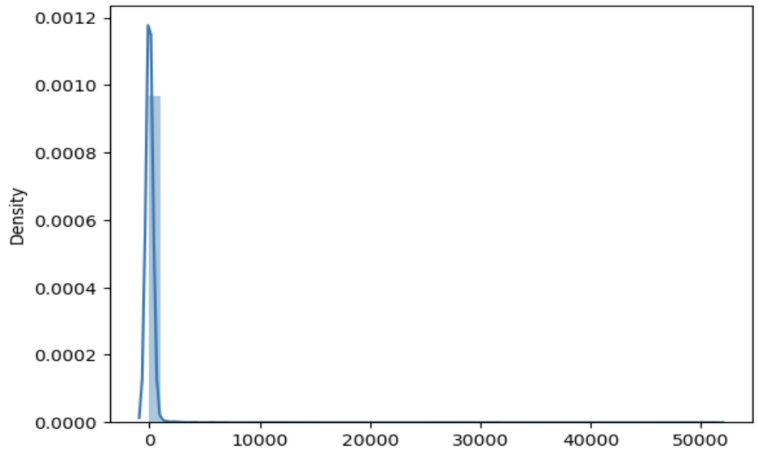


Figure 13: Density of our test set

As we can see, the density of our testing data and training data are skewed to the right.

How can we fix this? We need to do one more preprocessing step. We are going to use Standard Scalar transformation. Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. Why do we use it? It transforms the skewed data into a normal distribution it will bring mean to 0 and standard deviation to 1. The standard scaling is also sensitive towards outliers.

Using Standard Scalar from scikit learn, we get the following densities.

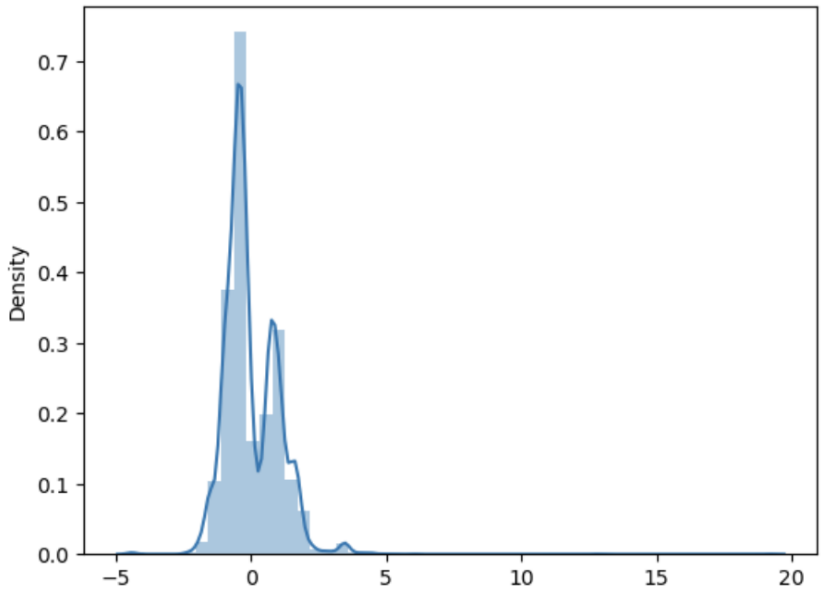


Figure 14: Density of scaled training set

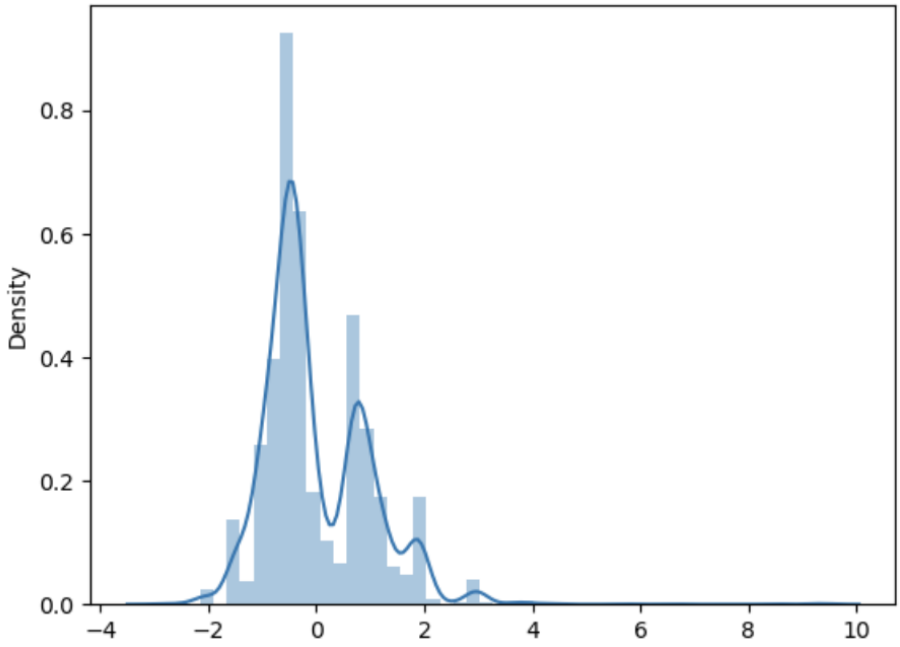


Figure 15: Density of scaled testing set

What we notice is that after the standard scaling the data is modified and the skewness is reduced.

Now we can model our dataset using four different algorithms: Logistic Regression, Random Forest Classifier, SVM and Grid Search.

III. Modeling Our Dataset

We have to note that predicting if a credit card application will be approved or not is a classification task. According to UCI, our dataset contains more instances that correspond to "Denied" status than instances corresponding to "Approved" status. Specifically, out of 690 instances, there are 383 (55.5%) applications that got denied and 307 (44.5%) applications that got approved. (We found out this in “Now What? Explore the Dataset” section of this report.)

Let’s apply this to our algorithms. For each of the following, we will evaluate our model on the test set with respect to classification accuracy. We will also take a look at each model’s confusion matrix. In the case of predicting credit card applications, it is equally important to see if our machine learning model is able to predict the approval status of the applications as denied that

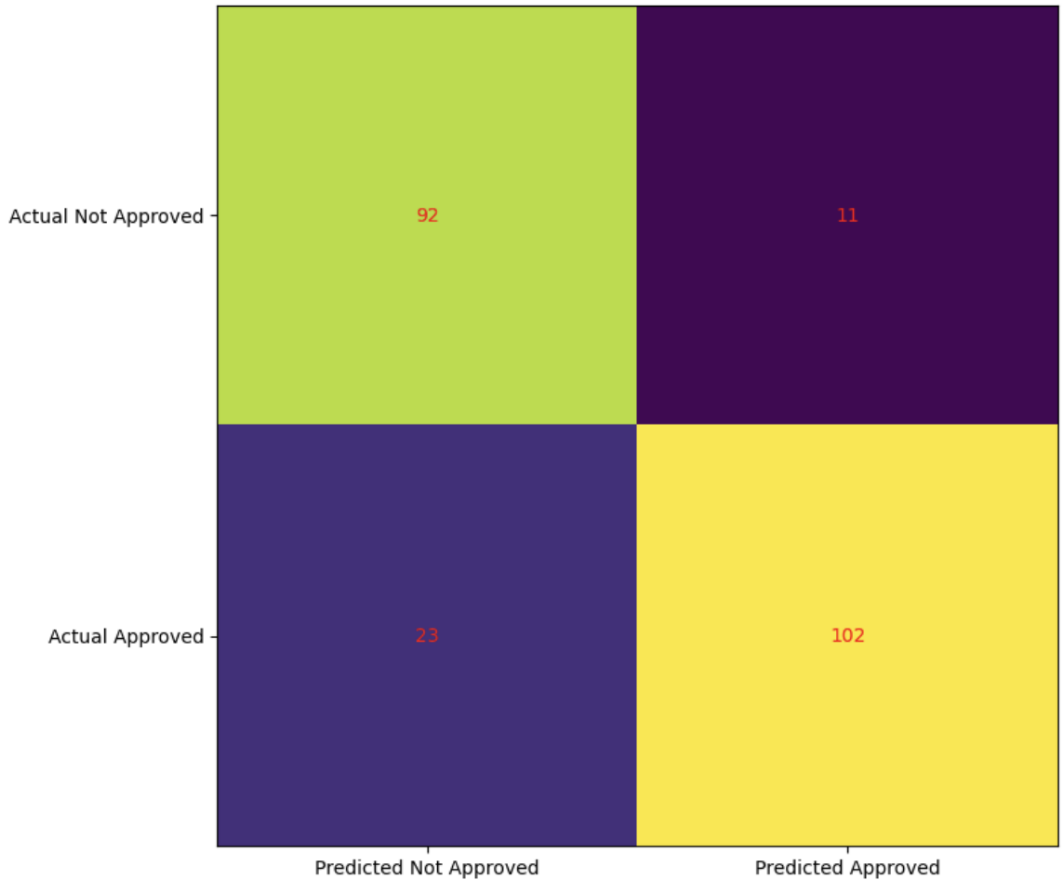
originally got denied. If our model is not performing well in this aspect, then it might end up approving the application that should have been approved. The confusion matrix helps us to view our model's performance from these aspects.

For the confusion matrix, the first element of the first row of the confusion matrix denotes the true negatives meaning the number of negative instances (denied applications) predicted by the model correctly. And the last element of the second row of the confusion matrix denotes the true positives meaning the number of positive instances (approved applications) predicted by the model correctly.

A. Logistic Regression

But how well does our model perform?

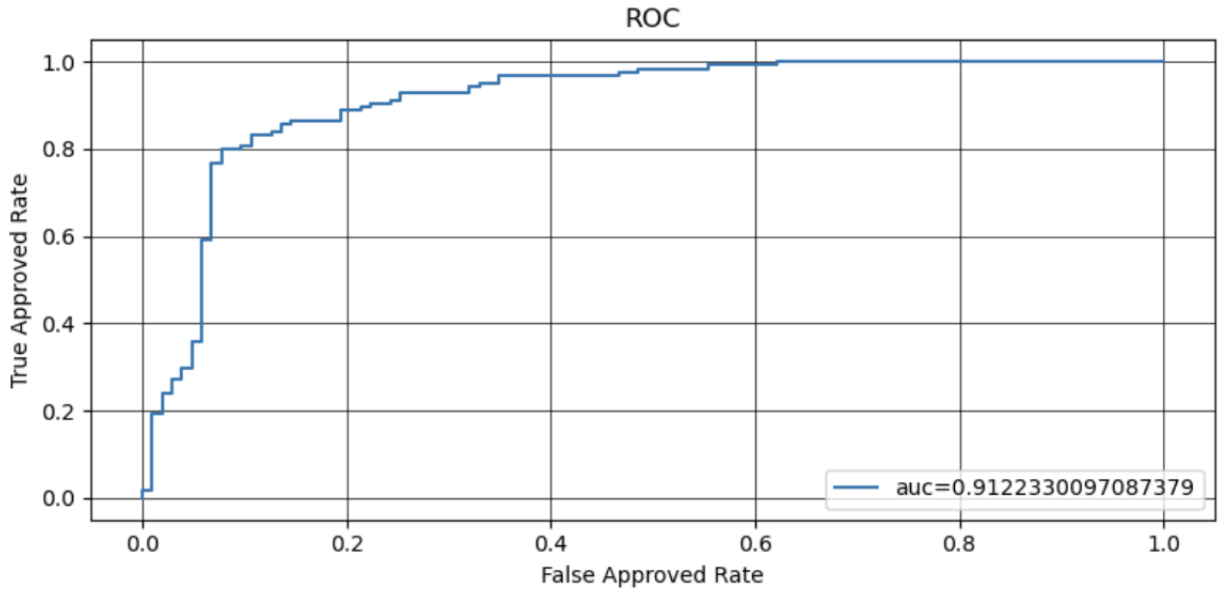
Confusion Matrix:



Classification report:

	precision	recall	f1-score	support
Not Approved	0.80	0.89	0.84	103
Approved	0.90	0.82	0.86	125
accuracy			0.85	228
macro avg	0.85	0.85	0.85	228
weighted avg	0.86	0.85	0.85	228

Receiver Operating Characteristic(ROC) curve:



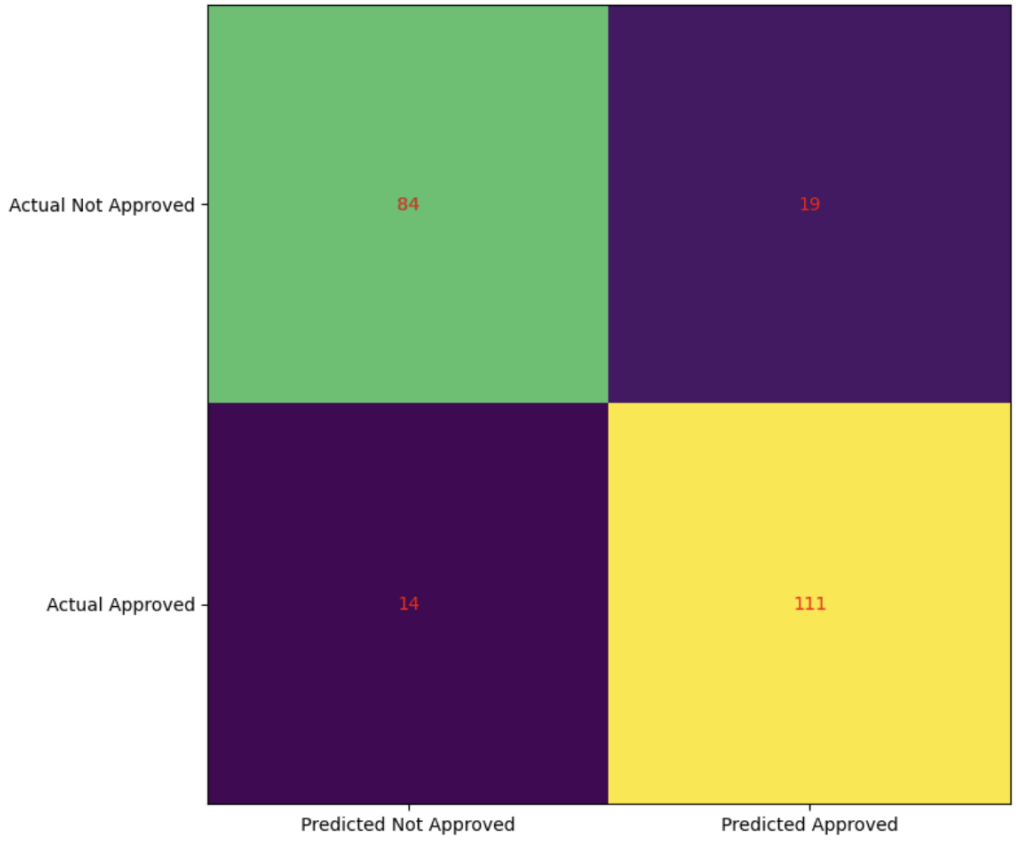
As we can see from our data using logistic regression, we can accuracy of 85.087%

Can we do better?

B. Random Forest Classifier

But how well does our model perform?

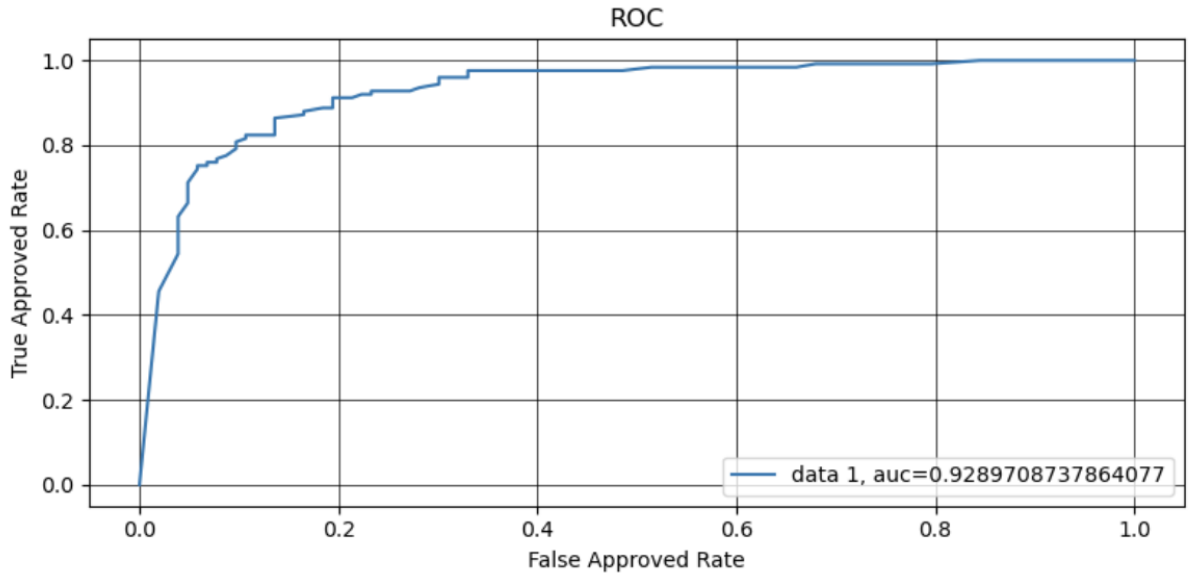
Confusion Matrix:



Classification Report:

	precision	recall	f1-score	support
Not Approved	0.86	0.82	0.84	103
Approved	0.85	0.89	0.87	125
accuracy			0.86	228
macro avg	0.86	0.85	0.85	228
weighted avg	0.86	0.86	0.85	228

Receiver Operating Characteristic(ROC) curve:



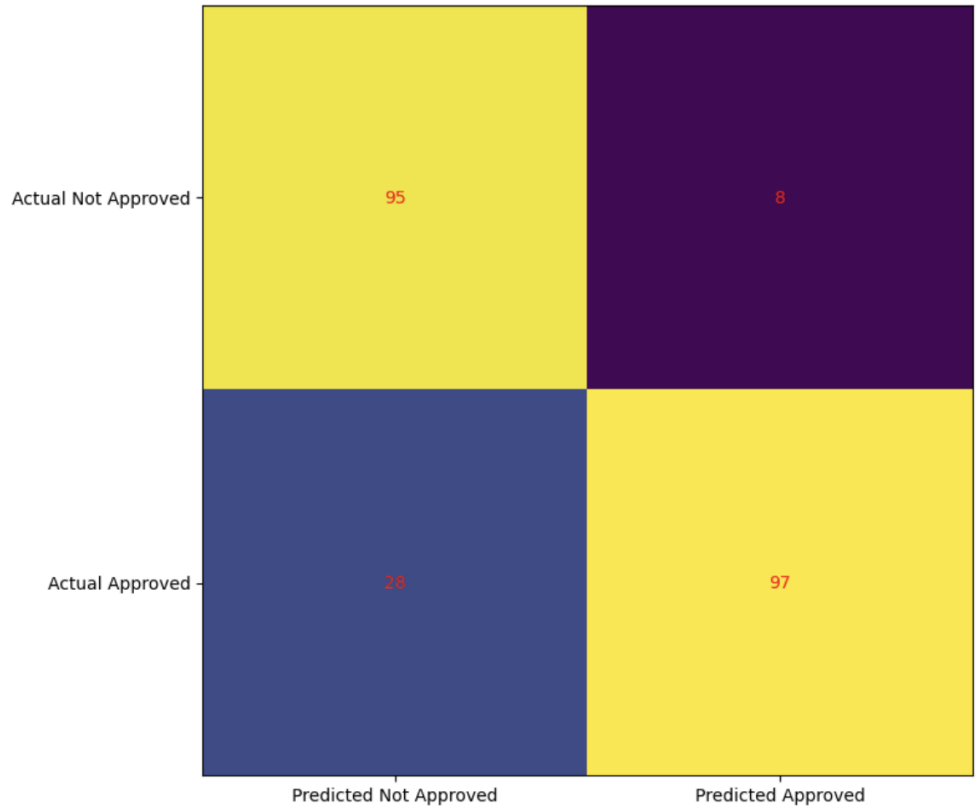
As we can see from our data using random forest classifier, we can accuracy of 85.526%

Slightly better than Logistic Regression. Can we do better?

C. Support Vector Machine

But how well does our model perform?

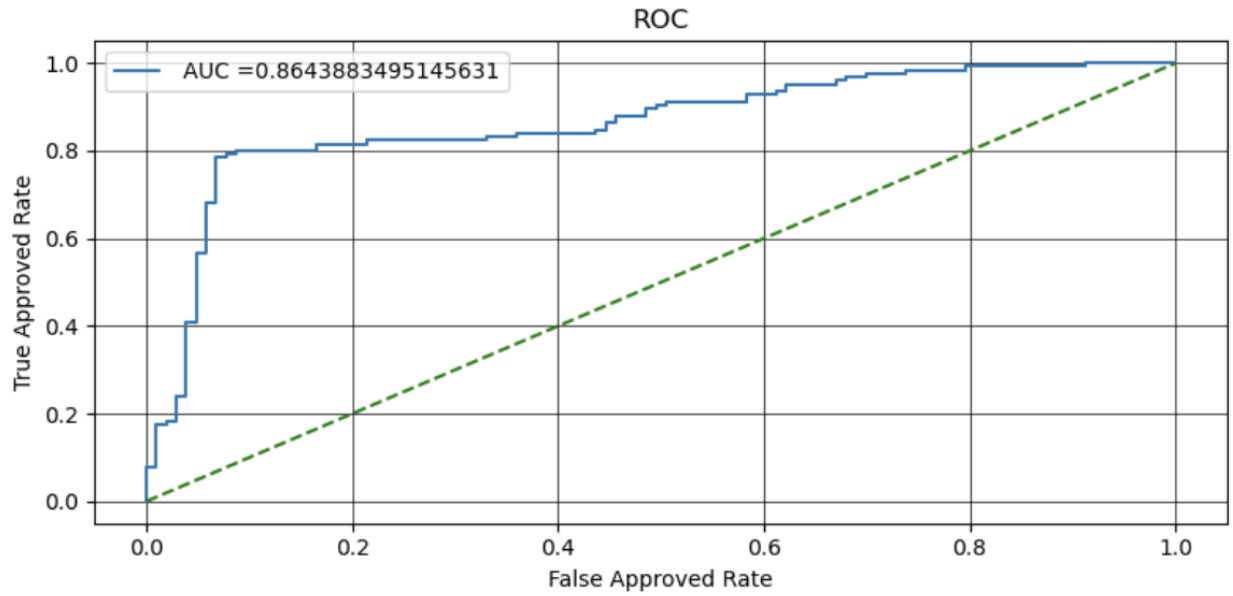
Confusion Matrix:



Classification Report:

	precision	recall	f1-score	support
Not Approved	0.77	0.92	0.84	103
Approved	0.92	0.78	0.84	125
accuracy			0.84	228
macro avg	0.85	0.85	0.84	228
weighted avg	0.86	0.84	0.84	228

Receiver Operating Characteristic(ROC) curve:



As we can see from our data using svm, we have an accuracy of 84.21% Uh oh, Logistic Regression and Random Forest Classifier worked better than SVM.

III. Conclusion

While building this credit card predictor, we tackled some of the most widely-known preprocessing steps such as scaling, label encoding, and missing value imputation. We finished with some machine learning to predict if a person's application for a credit card would get approved or not given some information about that person. We used machine learning algorithms such as: Logistic Regression, Random Forest Classifier, and Support Vector Machine. Let's compare our results. (See figure 16). For each algorithm we plot the Receiver Operating Characteristic curve. This curve plots the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity. The closer the AUC the perfect classifier and as it close to 0.5 it is considered a worthless classifier.

We see that Random Forest Classifier is best with an AUC of .928. This is amazing!

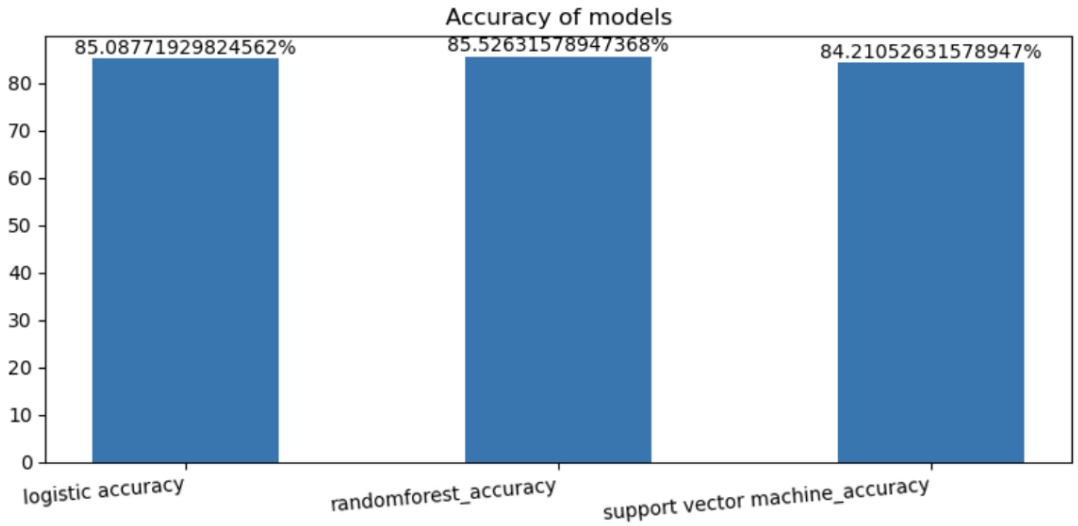


Figure 16.

As we can see from the chart above, Random Forest Classifier was more accurate than the rest of the machine learning algorithms. Key benefits of using Random Forest is that it can reduce risk of overfitting, provide flexibility, and it is easy to determine feature importance. Banks use Random Forest since it reduces time spent on data management and pre-processing tasks. It can be used to evaluate customers with high credit risk, to detect fraud, and also approval of credit card applications.

Citations

(1) Education, IBM Cloud. “What Is Random Forest?” *What Is Random Forest? | IBM*, 8 Dec. 2020, www.ibm.com/cloud/learn/random-forest.

(2) “Understanding Logistic Regression in Python.” *Understanding Logistic Regression in Python*, app.datacamp.com/workspace/w/11b0dd9d-49be-429d-992e-45f73b3c23fb. Accessed 7 Dec. 2022.

(3) Rane, Kiranmayi, et al. “Data Analysis on Credit Card Approval.” *Data Science Blog*, 2 July 2018, nycdatascience.com/blog/student-works/data-analysis-on-credit-card-approval.

(4) H Groenwold, Rolf H., and Olaf M. Dekkers. “Missing Data: The Impact of What Is Not There.” *Eje*, 1 Oct. 2020, eje.bioscientifica.com/view/journals/eje/183/4/EJE-20-0732.xml.

(5) Kumar, Ajitesh. “MinMaxScaler Vs StandardScaler - Python Examples - Data Analytics.” *Data Analytics*, 27 July 2020, vitalflux.com/minmaxscaler-standardscaler-python-examples.

(6) PyShark. “Label Encoding in Python - PyShark.” *PyShark*, 25 Mar. 2020, pyshark.com/label-encoding-in-python.

(7) “Simple Guide to Confusion Matrix Terminology.” *Data School*, 26 Mar. 2014, www.dataschool.io/simple-guide-to-confusion-matrix-terminology.

(8) Python, Real. “NumPy, SciPy, and Pandas: Correlation With Python – Real Python.” *NumPy, SciPy, and Pandas: Correlation With Python – Real Python*, realpython.com/numpy-scipy-pandas-correlation-python. Accessed 7 Dec. 2022.

(9) Kuhn, Ryan. “Analysis of Credit Approval Data.” *Analysis of Credit Approval Data*, rstudio-pubs-static.s3.amazonaws.com/73039_9946de135c0a49daa7a0a9eda4a67a72.html. Accessed 7 Dec. 2022.