Credit Card Approval Prediction System

Problem statement:

To predict whether an application for Credit card is approved or not based on gender, age, debt, married, bank customer, education level, ethnicity, no of years employed, prior default, employed, credit score, citizenship and income using **Logistic Regression** and **Decision Tree** and compare them**.**

Introduction:

A large number of credit card applications are received by commercial banks. Many of them are turned down for a variety of reasons, such as large debt balances, insufficient income, or too many inquiries on a person's credit report. Manually assessing these programmes is tedious, time-consuming, and error-prone. Fortunately, with the use of machine learning, this work can be automated, and almost every commercial bank does so nowadays. In this System, we'll use machine learning techniques to create an automatic credit card approval predictor.

Data set information:

LINK: http://archive.ics.uci.edu/ml/datasets/credit+approval

We find that since this data is confidential, the contributor of the dataset has anonymized the feature names to protect privacy.

- First, we will start off by loading and viewing the dataset.
- We will see that the dataset has a mixture of both numerical and non-numerical features, that it contains values from different ranges, plus that it contains a number of missing entries.
- We will have to preprocess the dataset to ensure the machine learning model we choose can make good predictions.
- After our data is in good shape, we will do some exploratory data analysis to build our intuitions.
- Finally, we will build a machine learning model that can predict if an individual's application for a credit card will be accepted

Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	Approved

b	30.83	0	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
a	58.67	4.46	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
a	24.50	0.5	u	g	q	h	1.5	t	f	0	f	g	00280	824	+
b	27.83	1.54	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+
b	32.08	4	u	g	m	v	2.5	t	f	0	t	g	00360	0	+
b	33.17	1.04	u	g	r	h	6.5	t	f	0	t	g	00164	31285	+
a	22.92	11.585	u	g	cc	v	0.04	t	f	0	f	g	00080	1349	+
b	54.42	0.5	y	p	k	h	3.96	t	f	0	f	g	00180	314	+
b	42.50	4.915	y	p	w	v	3.165	t	f	0	t	g	00052	1442	+
b	22.08	0.83	u	g	c	h	2.165	f	f	0	t	g	00128	0	+
b	29.92	1.835	u	g	c	h	4.335	t	f	0	f	g	00260	200	+
a	38.25	6	u	g	k	v	1	t	f	0	t	g	00000	0	+
b	48.08	6.04	u	g	k	v	0.04	f	f	0	f	g	00000	2690	+
a	45.	10.	u	g	q	v	5	t	t	7	t	g	000	0	+

	83	5										00			
b	36.67	4.415	y	p	k	v	0.25	t	t	10	t	g	00320	0	+
b	28.25	0.875	u	g	m	v	0.96	t	t	3	t	g	00396	0	+
a	23.25	5.875	u	g	q	v	3.17	t	t	10	f	g	00120	245	+
b	21.83	0.25	u	g	d	h	0.665	t	f	0	t	g	00000	0	+

Code and output:

```
# Import pandas
import pandas as pd

# Load dataset
cc_apps = pd.read_csv("/content/credit_card_approval.data")
cc_apps.columns=['Gender','Age','Debt','Married','BankCustomer','Education
Level','Ethnicity','YearsEmployed','PriorDefault','Employed','CreditScore'
,'DriversLicense','Citizen','ZipCode','Income','Approved']

# Print summary statistics
cc_apps_description = cc_apps.describe()
print(cc_apps_description)

print("\n")

# Print DataFrame information
cc_apps_info = cc_apps.info()
print(cc_apps_info)

print("\n")

# Inspect missing values in the dataset
print(cc_apps.tail(50))
```

```
# Import numpy
import numpy as np# Impute the missing values with mean imputation
cc_apps = cc_apps.fillna(cc_apps.mean())

# Count the number of NaNs in the dataset to verify
print(cc_apps.isnull().values.sum())


# Inspect missing values in the dataset
print(cc_apps.isnull().values.sum())

# Replace the '?'s with NaN
cc_apps = cc_apps.replace("?",np.NaN)

# Inspect the missing values again
print(cc_apps.tail(50))

# Iterate over each column of cc_apps

for col in cc_apps.columns:
    # Check if the column is of object type
    if cc_apps[col].dtypes == 'object':
        # Impute with the most frequent value
        cc_apps[col] =
cc_apps[col].fillna(cc_apps[col].value_counts().index[0])

# Count the number of NaNs in the dataset and print the counts to verify
print(cc_apps.isnull().values.sum())

# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Instantiate LabelEncoder
le = LabelEncoder()

# Iterate over all the values of each column and extract their dtypes
for col in cc_apps.columns:
    # Compare if the dtype is object
    if cc_apps[col].dtype=='object':
```

```
    # Use LabelEncoder to do the numeric transformation
        cc_apps[col]=le.fit_transform(cc_apps[col])
# Import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

# Drop features 10 and 13 and convert the DataFrame to a NumPy array
cc_apps = cc_apps.drop([cc_apps.columns[11],cc_apps.columns[13]], axis=1)
print(cc_apps)

cc_apps = cc_apps.values

# Segregate features and labels into separate variables
X,y = cc_apps[:,0:13], cc_apps[:,13]

# Instantiate MinMaxScaler and use it to rescale
scaler = MinMaxScaler(feature_range=(0,1))
rescaledX = scaler.fit_transform(X)
rescaledX

# Import train_test_split
from sklearn.model_selection import train_test_split

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(rescaledX,
                                                    y,
                                                    test_size=0.33,
                                                    random_state=42)
# Import LogisticRegression
from sklearn.linear_model import LogisticRegression

# Instantiate a LogisticRegression classifier with default parameter
values
logreg = LogisticRegression()

# Fit logreg to the train set
logreg.fit(X_train,y_train)

# Import confusion_matrix
from sklearn.metrics import confusion_matrix
# Use logreg to predict instances from the test set and store it
```

```
y_pred = logreg.predict(X_test)

# Get the accuracy score of logreg model and print it
print("Accuracy of logistic regression classifier: ", logreg.score(X_test,
y_test))
# Print the confusion matrix of the logreg model
confusion_matrix(y_test, y_pred)

import seaborn as sns
import matplotlib.pyplot as plt
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True,
cmap='Blues')

ax.set_title('Credit Card Approval Confusion Matrix by Logistic
Regression\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
model = DecisionTreeClassifier(max_depth=12,
                               min_samples_split=8,
                               random_state=1024)
model.fit(X_train, y_train)
y_predict = model.predict(X_test)

print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
print(pd.DataFrame(confusion_matrix(y_test,y_predict)))
ax = sns.heatmap(confusion_matrix(y_test, y_predict), annot=True,
cmap='Blues')

ax.set_title('Credit Card Approval Confusion Matrix by Decision
Tree\n\n');
```

```
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

!pip install colorama

from colorama import Fore, Back

gender=input("Are you male?\n\n")
age=input("\n\nWhat is your age?\n\n")
Debt=input("\n\nWhat is your Debt?\n\n")
Married=input("\n\nAre you married?\n\n")
BankCustomer=input("\n\nAre you a bank customer?\n\n")
EducationLevel=input("\n\nWhat is your Education Level?\n\n")
Ethnicity=input("\n\nWhat is your Ethnicity?\n\n")
YearsEmployed=input("\n\nHow many years were you employed?\n\n")
PriorDefault=input("\n\nDo you have any Prior defaults?\n\n")
Employed=input("\n\nAre you Employed?\n\n")
CreditScore=input("\n\nWhat is your Credit Score?\n\n")
Citizen=input("\n\nWhat is your Citizenship?\n\n")
Income=input("\n\nWhat is your Income?\n\n")
new_input=[[gender, age, Debt, Married,
BankCustomer,EducationLevel,Ethnicity,YearsEmployed,PriorDefault,Employed,
CreditScore,Citizen,Income]]

new_output=logreg.predict(new_input)
print(new_output)
print("By Logistic Regression")
if (new_output==1):
  print(Fore.WHITE, Back.GREEN +"Congratulations your credit card
application has been approved")
else:
  print(Fore.RED +"We regret to inform you that your credit card
application is declined.")
```

```
NEW_output=model.predict(new_input)
print(NEW_output)
print("By Decision Tree")
if (NEW_output==1):
  print(Fore.WHITE, Back.GREEN +"Congratulations your credit card
application has been approved")
else:
  print(Fore.RED +"We regret to inform you that your credit card
application is declined.")
```

Output:

```
             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


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Gender          690 non-null    object
 1   Age             690 non-null    object
 2   Debt            690 non-null    float64
 3   Married         690 non-null    object
 4   BankCustomer    690 non-null    object
 5   EducationLevel  690 non-null    object
 6   Ethnicity       690 non-null    object
 7   YearsEmployed   690 non-null    float64
 8   PriorDefault    690 non-null    object
 9   Employed        690 non-null    object
 10  CreditScore     690 non-null    int64
 11  DriversLicense  690 non-null    object
 12  Citizen         690 non-null    object
 13  ZipCode         690 non-null    object
 14  Income          690 non-null    int64
 15  Approved        690 non-null    object
dtypes: float64(2), int64(2), object(12)
memory usage: 86.4+ KB
None
```

	Gender	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	\
640	b	34.17	2.750	u	g	i	bb	
641	?	33.17	2.250	y	p	cc	v	
642	b	31.58	0.750	y	p	aa	v	
643	a	52.50	7.000	u	g	aa	h	
644	b	36.17	0.420	y	p	w	v	
645	b	37.33	2.665	u	g	cc	v	
646	a	20.83	8.500	u	g	c	v	
647	b	24.08	9.000	u	g	aa	v	
648	b	25.58	0.335	u	g	k	h	
649	a	35.17	3.750	u	g	ff	ff	
650	b	48.08	3.750	u	g	i	bb	
651	a	15.83	7.625	u	g	q	v	
652	a	22.50	0.415	u	g	i	v	
653	b	21.50	11.500	u	g	i	v	
654	a	23.58	0.830	u	g	q	v	
655	a	21.08	5.000	y	p	ff	ff	
656	b	25.67	3.250	u	g	c	h	
657	a	38.92	1.665	u	g	aa	v	
658	a	15.75	0.375	u	g	c	v	
659	a	28.58	3.750	u	g	c	v	
660	b	22.25	9.000	u	g	aa	v	
661	b	29.83	3.500	u	g	c	v	
662	a	23.50	1.500	u	g	w	v	
663	b	32.08	4.000	y	p	cc	v	
664	b	31.08	1.500	y	p	w	v	
665	b	31.83	0.040	y	p	m	v	
666	a	21.75	11.750	u	g	c	v	
667	a	17.92	0.540	u	g	c	v	
668	b	30.33	0.500	u	g	d	h	
669	b	51.83	2.040	y	p	ff	ff	
670	b	47.17	5.835	u	g	w	v	
671	b	25.83	12.835	u	g	cc	v	
672	a	50.25	0.835	u	g	aa	v	
673	?	29.50	2.000	y	p	e	h	

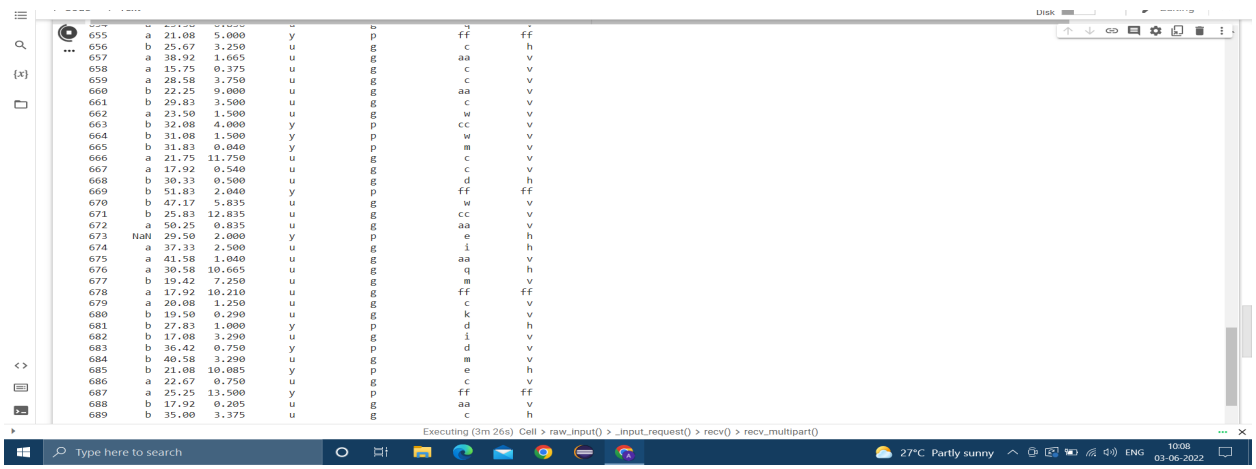
674	a	37.33	2.500	u	g	i	h	
675	a	41.58	1.040	u	g	aa	v	
676	a	30.58	10.665	u	g	q	h	
677	b	19.42	7.250	u	g	m	v	
678	a	17.92	10.210	u	g	ff	ff	
679	a	20.08	1.250	u	g	c	v	
680	b	19.50	0.290	u	g	k	v	
681	b	27.83	1.000	y	p	d	h	
682	b	17.08	3.290	u	g	i	v	
683	b	36.42	0.750	y	p	d	v	
684	b	40.58	3.290	u	g	m	v	
685	b	21.08	10.085	y	p	e	h	
686	a	22.67	0.750	u	g	c	v	
687	a	25.25	13.500	y	p	ff	ff	
688	b	17.92	0.205	u	g	aa	v	
689	b	35.00	3.375	u	g	c	h	

	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	\
640	2.500	f	f	0	t	g	
641	3.500	f	f	0	t	g	
642	3.500	f	f	0	t	g	
643	3.000	f	f	0	f	g	
644	0.290	f	f	0	t	g	
645	0.165	f	f	0	t	g	
646	0.165	f	f	0	f	g	
647	0.250	f	f	0	t	g	
648	3.500	f	f	0	t	g	
649	0.000	f	t	6	f	g	
650	1.000	f	f	0	f	g	
651	0.125	f	t	1	t	g	
652	0.335	f	f	0	t	s	
653	0.500	t	f	0	t	g	
654	0.415	f	t	1	t	g	
655	0.000	f	f	0	f	g	
656	2.290	f	t	1	t	g	

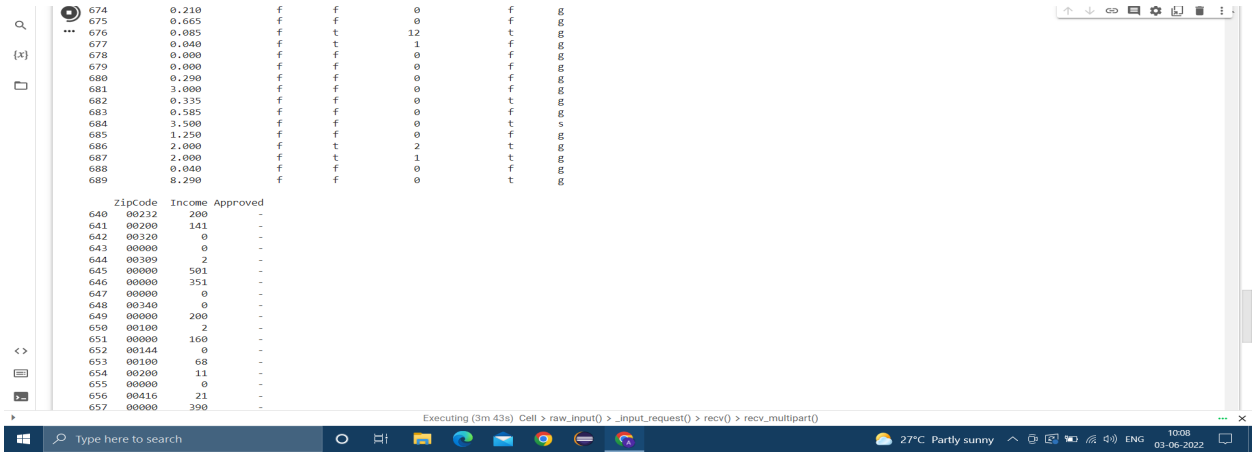
657	0.250	f	f	0	f	g
658	1.000	f	f	0	f	g
659	0.250	f	t	1	t	g
660	0.085	f	f	0	f	g
661	0.165	f	f	0	f	g
662	0.875	f	f	0	t	g
663	1.500	f	f	0	t	g
664	0.040	f	f	0	f	s
665	0.040	f	f	0	f	g
666	0.250	f	f	0	t	g
667	1.750	f	t	1	t	g
668	0.085	f	f	0	t	s
669	1.500	f	f	0	f	g
670	5.500	f	f	0	f	g
671	0.500	f	f	0	f	g
672	0.500	f	f	0	t	g
673	2.000	f	f	0	f	g
674	0.210	f	f	0	f	g
675	0.665	f	f	0	f	g
676	0.085	f	t	12	t	g
677	0.040	f	t	1	f	g
678	0.000	f	f	0	f	g
679	0.000	f	f	0	f	g
680	0.290	f	f	0	f	g
681	3.000	f	f	0	f	g
682	0.335	f	f	0	t	g
683	0.585	f	f	0	f	g
684	3.500	f	f	0	t	s
685	1.250	f	f	0	f	g
686	2.000	f	t	2	t	g
687	2.000	f	t	1	t	g
688	0.040	f	f	0	f	g
689	8.290	f	f	0	t	g

	ZipCode	Income	Approved
640	00232	200	-
641	00200	141	-
642	00320	0	-
643	00000	0	-
644	00309	2	-
645	00000	501	-
646	00000	351	-
647	00000	0	-
648	00340	0	-
649	00000	200	-
650	00100	2	-
651	00000	160	-
652	00144	0	-
653	00100	68	-
654	00200	11	-
655	00000	0	-
656	00416	21	-
657	00000	390	-
658	00120	18	-
659	00040	154	-
660	00000	0	-
661	00216	0	-
662	00160	0	-
663	00120	0	-
664	00160	0	-
665	00000	0	-
666	00180	0	-
667	00080	5	-
668	00252	0	-
669	00120	1	-
670	00465	150	-
671	00000	2	-
672	00240	117	-
673	00256	17	-

```
674   00260     246        -
675   00240     237        -
676   00129       3        -
677   00100       1        -
678   00000      50        -
679   00000       0        -
680   00280     364        -
681   00176     537        -
682   00140       2        -
683   00240       3        -
684   00400       0        -
685   00260       0        -
686   00200     394        -
687   00200       1        -
688   00280     750        -
689   00000       0        -
0
0
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:26: FutureWarning: Dropping of nuisanc
    Gender    Age    Debt Married BankCustomer EducationLevel Ethnicity  \
640      b  34.17   2.750       u            g              i        bb
641    NaN  33.17   2.250       y            p             cc         v
642      b  31.58   0.750       y            p             aa         v
643      a  52.50   7.000       u            g             aa         h
644      b  36.17   0.420       y            p              w         v
645      b  37.33   2.665       u            g             cc         v
646      a  20.83   8.500       u            g              c         v
647      b  24.08   9.000       u            g             aa         v
648      b  25.58   0.335       u            g              k         h
649      a  35.17   3.750       u            g             ff        ff
650      b  48.08   3.750       u            g              i        bb
651      a  15.83   7.625       u            g              q         v
652      a  22.50   0.415       u            g              i         v
653      b  21.50  11.500       u            g              i         v
654      a  23.58   0.830       u            g              q         v
655      a  21.08   5.000       y            p             ff        ff
```



	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	\
640	2.500	f	f	0	t	g	
641	3.500	f	f	0	t	g	
642	3.500	f	f	0	t	g	
643	3.000	f	f	0	f	g	
644	0.290	f	f	0	t	g	
645	0.165	f	f	0	t	g	
646	0.165	f	f	0	f	g	
647	0.250	f	f	0	t	g	
648	3.500	f	f	0	t	g	
649	0.000	f	t	6	f	g	
650	1.000	f	f	0	f	g	
651	0.125	f	t	1	t	g	
652	0.335	f	f	0	t	s	
653	0.500	t	f	0	t	g	
654	0.415	f	t	1	t	g	
655	0.000	f	f	0	f	g	
656	2.290	f	t	1	t	g	
657	0.250	f	f	0	f	g	
658	1.000	f	f	0	f	g	
659	0.250	f	t	1	t	g	
660	0.085	f	f	0	f	g	
661	0.165	f	f	0	f	g	
662	0.875	f	f	0	t	g	
663	1.500	f	f	0	t	g	
664	0.040	f	f	0	f	s	
665	0.040	f	f	0	f	g	
666	0.250	f	f	0	t	g	
667	1.750	f	t	1	t	g	
668	0.085	f	f	0	t	s	
669	1.500	f	f	0	f	g	
670	5.500	f	f	0	f	g	
671	0.500	f	f	0	f	g	
672	0.500	f	f	0	t	g	
673	2.000	f	f	0	f	g	



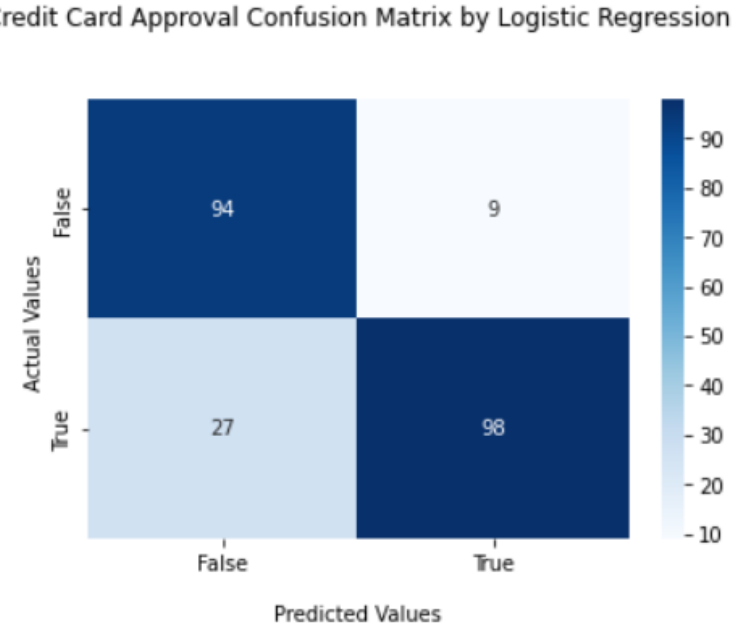
674	0.210	f	f	0	f	g	
675	0.665	f	f	0	f	g	
676	0.085	f	t	12	t	g	
677	0.040	f	t	1	f	g	
678	0.000	f	f	0	f	g	
679	0.000	f	f	0	f	g	
680	0.290	f	f	0	f	g	
681	3.000	f	f	0	f	g	
682	0.335	f	f	0	t	g	
683	0.585	f	f	0	f	g	
684	3.500	f	f	0	t	s	
685	1.250	f	f	0	f	g	
686	2.000	f	t	2	t	g	
687	2.000	f	t	1	t	g	
688	0.040	f	f	0	f	g	
689	8.290	f	f	0	t	g	

	ZipCode	Income	Approved
640	00232	200	-
641	00200	141	-
642	00320	0	-
643	00000	0	-
644	00309	2	-
645	00000	501	-
646	00000	351	-
647	00000	0	-
648	00340	0	-
649	00000	200	-
650	00100	2	-
651	00000	160	-
652	00144	0	-
653	00100	68	-
654	00200	11	-
655	00000	0	-
656	00416	21	-
657	00000	390	-

658	00120	18	-
659	00040	154	-
660	00000	0	-
661	00216	0	-
662	00160	0	-
663	00120	0	-
664	00160	0	-
665	00000	0	-
666	00180	0	-
667	00080	5	-
668	00252	0	-
669	00120	1	-
670	00465	150	-
671	00000	2	-
672	00240	117	-
673	00256	17	-
674	00260	246	-
675	00240	237	-
676	00129	3	-
677	00100	1	-
678	00000	50	-
679	00000	0	-
680	00280	364	-
681	00176	537	-
682	00140	2	-
683	00240	3	-
684	00400	0	-
685	00260	0	-
686	00200	394	-
687	00200	1	-
688	00280	750	-
689	00000	0	-

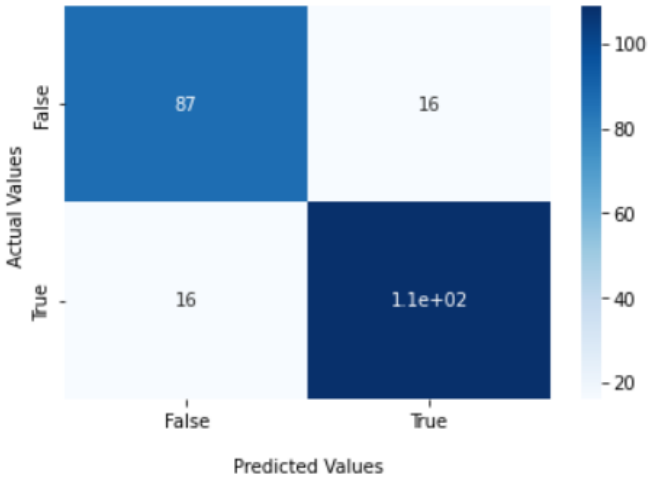
0

```
[690 rows x 14 columns]
Accuracy of logistic regression classifier:  0.8421052631578947
[[94  9]
 [27 98]]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:26: FutureWarning: Droppi
```



```
Accuracy Score is 0.85965
    0    1
0  87   16
1  16  109
```

Credit Card Approval Confusion Matrix by Decision Tree



Predicted Values

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting colorama
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Installing collected packages: colorama
Successfully installed colorama-0.4.4
Are you male?

1


What is your age?

0


What is your Debt?

0


Are you married?

0


Are you a bank customer?

0
```

```
What is your Education Level?

0


What is your Ethnicity?

0


How many years were you employed?

0


Do you have any Prior defaults?

0


Are you Employed?

0


What is your Credit Score?

0
```

```
What is your Citizenship?

0


What is your Income?

0
[1.]
By Logistic Regression
 Congratulations your credit card application has been approved
[1.]
By Decision Tree
 Congratulations your credit card application has been approved
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:566: FutureWarning: Arrays of bytes/strings is being converted to
  X = check_array(X, **check_params)
```

Conclusion:

After preprocessing the given dataset to add missing values by mean imputation, label encoding non numeric values into numeric, scaling the dataset and training, testing the dataset to create a model that predicts whether the application is approved or not after taking inputs from the user. We conclude that we have implemented this system using Logistic Regression and Decision Tree.

The Accuracy of this program using Logistic Regression is 84%.
The Accuracy of this program using the Decision Tree is 85%.

References:

https://www.geeksforgeeks.org/understanding-logistic-regression/
https://www.geeksforgeeks.org/decision-tree-introduction-example/
https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/
http://rstudio-pubs-static.s3.amazonaws.com/73039_9946de135c0a49daa7a0a9eda4a67a72.html#classification-and-regression-tree