### Applying Machine Learning To Financial Risk Management Using IBM Watson

**A PROJECT REPORT**

*submitted in partial fulfilment of the requirements*

*for the Certification of*

### Applied Data Science

*by*

Anvesh Kumar Deo

Abhishek Saha

Anil R.Shanbhag

Bhuvnesh Birla

*under the guidance of*

**

*July 2021*

**Introduction**

1. **Overview**

Timely loan repayment along with determining the level of customer reliability is one of the major elements of credit risk assessment. Based on the customer's characteristics credit history analysis and scoring is done. Credit scoring is one of the methods widely used for estimation of the risks associated in granting a loan, or rather the probability of its non-repayment. On the basis of the calculation according to data provided in the loan applications the customer score is obtained which determines the risk levels associated with it. Regardless of how it is calculated and what characteristics taken into account, it eliminates the human factors, adds objectivity to the process which in turn speeds up the process makes it smooth and reduce the risk. Banks usually sanction the loan on the basis of qualitative and quantitative analysis.  Based on statistical methods credit scoring helps to predict the probability of a certain events occurring in the future. Credit Scoring models can be classified according to different criteria. Thus, we can talk about a scoring of individuals or companies or credit card, cash or mortgage scoring. The goal of the scoring models for most of the parts is to determine the risk of debt default.

1. **Purpose**

Many lenders and financial institutions use statistical credit scoring analysis to determine the creditworthiness of a person or small scale business. Creditworthiness is how a lender will determines the failure of the debt on debt obligation and whether the person is suitable for new credit or not. Lenders uses credit scoring for the analysis of the risks associated with the sanction of the loan or credit, or to decide whether to extend or deny the credit. Credit scoring determines the person’s ability to borrow money for mortgages and for different purpose.

**Literature Review**

1. **Existing Problem:**

## [Financial Risks and Management](https://www.sciencedirect.com/science/article/pii/B9780081022955100654) :

Ekaterina Svetlova, Karl-Heinz Thielmann

Traditional Banking methods have limitations where they cannot analyse large volumes of data and are entirely dependent on credit scores and limited values. These greatly reduce the capability of institutions to reduce risk.

The number of errors that take place on a daily basis in traditional systems is a cause for concern. The cost and manpower required to run this system are immense and it takes a large amount of time for any changes to be implemented in the system.

These concepts are explored in great detail in this particular literature.

1. **Proposed Solutions:**

Palgrave Studies in Digital Business & Enabling Technologies

Chapter 3: Machine Learning and AI for Risk Management- Saqib Aziz and Michael Dowling

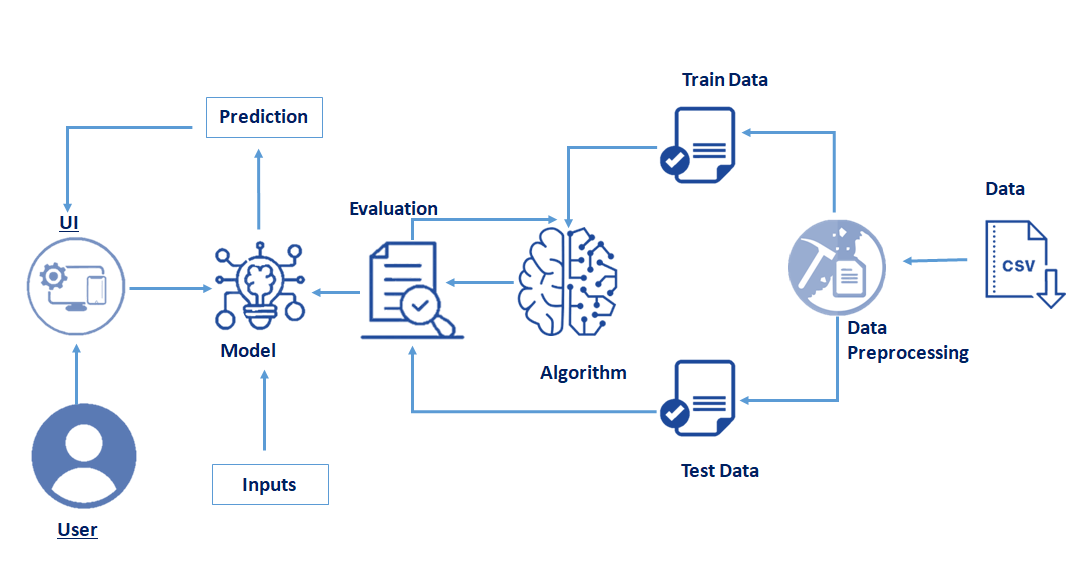
This particular chapter explores the possible uses of AI and Machine Learning in well established banking systems, it explores the possible use of ML to free man power and reduce the various risks prevalent in the banking system.

Machine Learning algorithms give investment managers, consumers and entire banking corporations future insights into the how the market will change much earlier than traditional banking models.

Using machine learning techniques, banks and financial institutions can significantly lower the risk levels by analysing a massive volume of data sources. Unlike the traditional methods which are usually limited to essential information such as credit score, ML can analyze significant volumes of personal information to reduce their risk.

**Theoretical Analysis**

1. **Block Diagram**



1. **Hardware/Software designing**

* Initially, we have used Jupyter Notebook as a tool for data retrieving and preprocessing of the dataset and later on moved towards the model building by applying various classification algorithms.
* After performing all the algorithms that comes under classification, we have chosen the one that has given the highest accuracy and implemented it on a web application using the flask framework for server-side scripting.
* With the help of flask a user can interact with the UI (User Interface) to enter the input values that can be analyzed by the model based on which prediction is showcased on the UI.
* In order to make our model deployable from any computer or system we have uploaded and trained our model on IBM Watson Studio (which enables deployment of models by using IBM cloud, ML engine and cloud storage facilities of IBM).

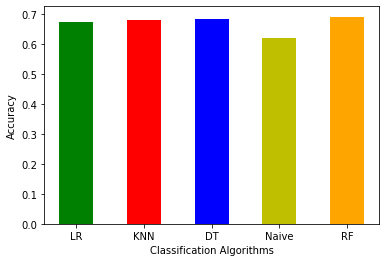
**Experimental Investigations**

In order to decide on the classification algorithm we decided to test each one of the classification algorithms on the given test set and thereby compare their accuracies to finally decide which is the best one.

The classification algorithms along with their respective accuracies have been listed in the table below:

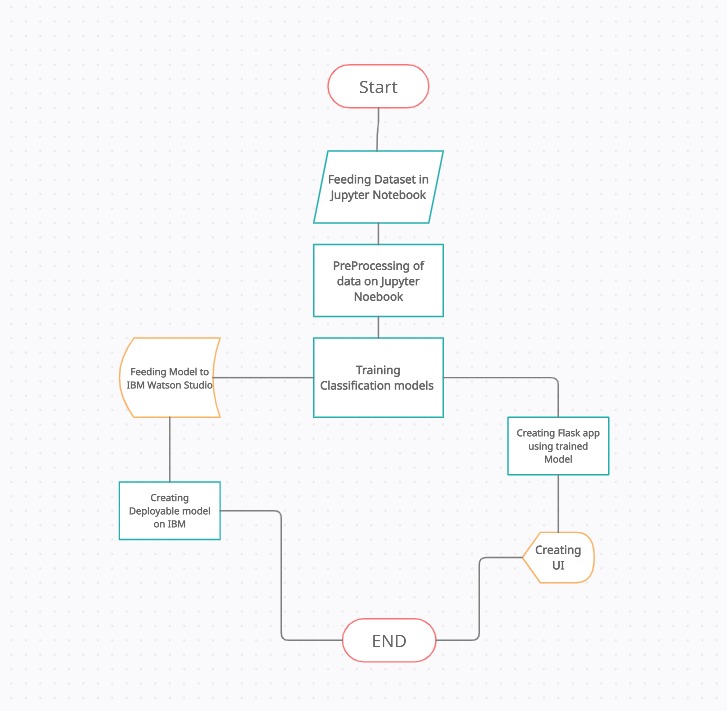
|  |  |  |
| --- | --- | --- |
| Sl No | Classification Algorithm | Accuracy |
| 1. | Logistic Regression | 0.672 |
| 2. | KNN | 0.678 |
| 3. | Decision Tree | 0.681 |
| 4. | Naïve Bayes | 0.621 |
| 5. | Random Forest | 0.690 |

Graphical Representation of the above table:



After considering the above results we used the Random Forest Classifier to build our model.

**Flow Chart**

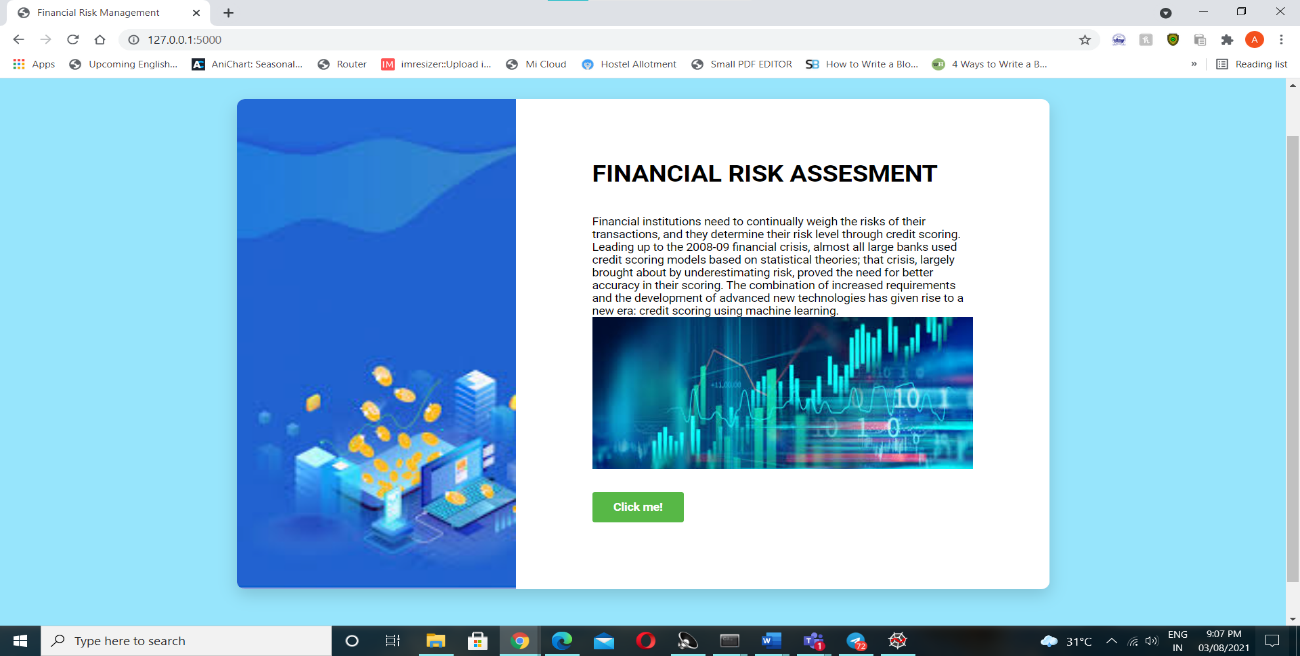


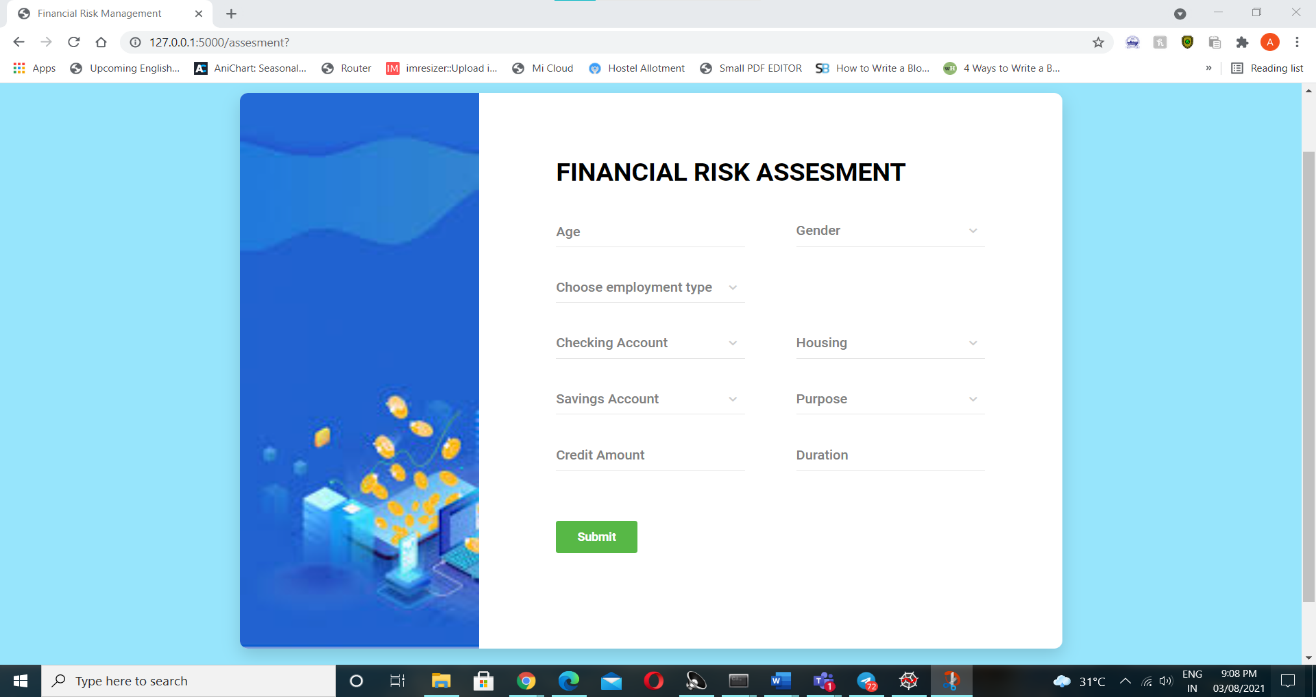
**Results**

The model we selected for Financial Risk Management is Random Forest Classifier.

Accuracy of the model is 70%.

We used the German Credit database to train and test our model.

**Output:**



**Advantages:**

* Scalability: The ML model can easily be scaled up to meet the requirement.
* It is quite cost effective as it requires very little capital and can take decisions much faster.
* The ML model can easily be changed or modified to accommodate changes in the market.
* The automatization of credit risk testing ultimately reduces losses for banks.

**Disadvantages:**

* When using an algorithm, it can be difficult to trust the decision as there is less transparency.
* Even with the best algorithm, there are still many human elements to take into consideration.
* The effectiveness of an algorithm can reduce with changes in economic factors and will constantly need to be updated and reviewed.

**Applications**

Our model can help financial institutions determine the credit worthiness of potential customers. By analyzing past spending behaviour and patterns, a system could identify how much credit should be extended to a given customer. The technology would be especially useful in the case of new customers or those who lack a long credit history, i.e. millennials. Automating credit and risk scoring processes on a mass scale can help banks enhance their credit and risk scoring models across the board.

**Conclusion**

The main objective of this project was to build machine learning algorithms that would be able to identify potential defaulters and therefore reduce company loss. The best model possible would be the one that could minimize false negatives, identifying all defaulters among the client base, while also minimizing false positives, preventing clients to be wrongly classified as defaulters.

Meeting these requirements can be quite difficult as there is a trade-off between precision and recall, meaning that increasing the value of one of these metrics often decreases the value of the other. Considering the importance of minimizing company loss, we decided to give more emphasis on reducing false positives, searching for the best hyperparameters that could increase the recall rate.

**Future Scope**

For accepting or rejecting a loan, a credit scoring model is a tool which is used for decision-making. A credit scoring model is the classification statistical model which is based on the borrower's information, it allows one to distinguish between “good” or “bad” loans. It is just one of the factors used for evaluating a credit application of the borrowers.

Although credit scoring methods are linked applications in banking and finance, they can be widely used in a large variety of other data analytics problems, such as:

 Factors which can contribute to a consumer’s choice?

What are some important factors which can generate the biggest impact to a consumer’s choice?

With a further boost in each of the impact factors what profit can be associated?

Will customer be willing to adopt a new service?

Such questions can all be answered within the same classification statistical Model.

**Bibliography**

* Exploring the potential of machine learning: How machine learning can support financial risk management

- Ohlsson, Caroline

Uppsala University, Disciplinary Domain of Humanities and Social Sciences, Faculty of Social Sciences, Department of Business Studies.

* Interaction between financial risk measures and machine learning methods.

- Jun-ya Gotoh, Akiko Takeda & Rei Yamamoto

# Machine Learning in Banking Risk Management

by [Martin Leo](https://sciprofiles.com/profile/535550), [Suneel Sharma](https://sciprofiles.com/profile/855200) and [K. Maddulety](https://sciprofiles.com/profile/652449)

SP Jain School of Global Management, Sydney 2127, Australia

* Machine Learning: A Revolution in Risk Management and Compliance?

Bart van Liebergen – Associate Policy Advisor, Institute of International Finance

**APPENDIX**

1. **Source Code:**

1.**Jupyter Notebook:**

import pandas as pd  
import numpy as np #Importing the required libraries for the project  
import matplotlib.pyplot as plt  
import joblib

df=pd.read\_csv('german\_credit\_data.csv') #Importing the dataset from local drive   
df

dataset=pd.read\_csv('german\_credit\_data.csv')

df.columns

cols=['Sex','Housing', 'Saving accounts', 'Checking account', 'Purpose']  
for i in cols:  
 print(df[i].value\_counts()) #Checking for types of values in the Dataset columns

#checking for null values

df.describe()

df.info()

df.isnull().any() #Checking for null values

import seaborn as sns

sns.pairplot(df)

sns.distplot(df['Duration'].dropna())

df['Saving accounts'].fillna(df['Saving accounts'].mode().iloc[0],inplace=True)

#Taking care of categorical null values in Saving accounts   
df

df['Checking account'].fillna(df['Checking account'].mode().iloc[0],inplace=True)

#Taking care of categorical null values in Checking account

df.isnull().any() #Verifying removal of null data

df['Job'].value\_counts()

df['Housing'].value\_counts()

df['Saving accounts'].value\_counts()

df['Purpose'].value\_counts()

df['Checking account'].value\_counts()

from sklearn.compose import ColumnTransformer  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.preprocessing import LabelEncoder  
lb=LabelEncoder()

df["Risk"]=lb.fit\_transform(df["Risk"])

df

ct=ColumnTransformer([("on",OneHotEncoder(),[1,3,4,5,8])],remainder='passthrough')

df=ct.fit\_transform(df)  
df

joblib.dump(ct,'onehot.save') # Saving the column transformation

df.shape

from scipy import stats

z=np.abs(stats.zscore(df))  
z

threshold=3  
np.where(z>threshold)

df\_no\_outliers=df[(z<=3).all(axis=1)]  
df\_no\_outliers

x=df[:,:-1]  
x

y=df[:,-1]  
y

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()  
x=sc.fit\_transform(x)  
x

joblib.dump(sc,'Scalar.save') #Saving Scalar transformation

['Scalar.save']

from sklearn.model\_selection import train\_test\_split  
x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.33,random\_state=45)

# Applying different Classification Algorithms

from sklearn.linear\_model import LogisticRegression

lr=LogisticRegression()

lr.fit(x\_train,y\_train)

LogisticRegression()

y\_pred=lr.predict(x\_test)  
y\_pred

y\_test

from sklearn.metrics import accuracy\_score  
lr\_accuracy=accuracy\_score(y\_test,y\_pred)  
lr\_accuracy

from sklearn.model\_selection import GridSearchCV

from sklearn.neighbors import KNeighborsClassifier

KNeighborsClassifier()

KNeighborsClassifier()

knn\_grid=GridSearchCV(estimator=KNeighborsClassifier(),param\_grid={'n\_neighbors':np.arange(1,20)},cv=5)

knn\_grid.fit(x\_train,y\_train)

GridSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_grid={'n\_neighbors': array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,  
 18, 19])})

knn\_grid.best\_params\_

knn=KNeighborsClassifier(n\_neighbors=15)  
knn.fit(x\_train,y\_train)

KNeighborsClassifier(n\_neighbors=15)

y\_pred=knn.predict(x\_test)  
y\_pred

from sklearn.metrics import accuracy\_score  
knn\_accuracy=accuracy\_score(y\_test,y\_pred)  
knn\_accuracy

from sklearn.tree import DecisionTreeClassifier

dt\_grid= GridSearchCV(estimator=DecisionTreeClassifier(),param\_grid={'criterion':['gini','entropy'],'max\_depth':np.arange(2,7)},cv=5)

dt\_grid.fit(x\_train,y\_train)

GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),  
 param\_grid={'criterion': ['gini', 'entropy'],  
 'max\_depth': array([2, 3, 4, 5, 6])})

dt\_grid.best\_params\_

dt=DecisionTreeClassifier(criterion='gini',max\_depth=2)

dt.fit(x\_train,y\_train)

DecisionTreeClassifier(max\_depth=2)

y\_pred=dt.predict(x\_test)  
y\_pred

from sklearn.metrics import accuracy\_score  
dt\_accuracy=accuracy\_score(y\_test,y\_pred)  
dt\_accuracy

from six import StringIO  
from IPython.display import Image  
from sklearn.tree import export\_graphviz  
import pydotplus  
dot\_data=StringIO()  
export\_graphviz(dt,out\_file=dot\_data,filled=True,rounded=True,special\_characters=True)  
graph=pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
Image(graph.create\_png())

from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.naive\_bayes import GaussianNB

pipe=Pipeline([("mn",MinMaxScaler()),("naive",GaussianNB())])

pipe.fit(x\_train,y\_train)

Pipeline(steps=[('mn', MinMaxScaler()), ('naive', GaussianNB())])

y\_pred=pipe.predict(x\_test)  
y\_pred

from sklearn.metrics import accuracy\_score  
naive\_accuracy=accuracy\_score(y\_test,y\_pred)  
naive\_accuracy

from sklearn.model\_selection import GridSearchCV  
from sklearn.ensemble import RandomForestClassifier

RandomForestClassifier()

RandomForestClassifier()

rf\_grid=GridSearchCV(estimator=RandomForestClassifier(),param\_grid={'n\_estimators':np.arange(1,50),'criterion':['gini','entropy'],'max\_depth':np.arange(2,10)},cv=5)

rf\_grid.fit(x\_train,y\_train)

rf\_grid.best\_params\_

rf=RandomForestClassifier(criterion='gini',max\_depth=6,n\_estimators= 37)

rf.fit(x\_train,y\_train)

RandomForestClassifier(max\_depth=6, n\_estimators=37)

y\_pred=rf.predict(x\_test)  
y\_pred

from sklearn.metrics import accuracy\_score  
rf\_accuracy=accuracy\_score(y\_test,y\_pred) #Accuracy is highest in random forest classification therefore we are utilising it in model making  
rf\_accuracy

from sklearn.metrics import confusion\_matrix  
cm = confusion\_matrix(y\_test,y\_pred)  
cm

import sklearn.metrics as metrics

fpr,tpr,threshold=metrics.roc\_curve(y\_test,y\_pred)

roc\_auc=metrics.auc(fpr,tpr)  
roc\_auc

plt.plot(fpr,tpr,label='AUC=%0.2f'%roc\_auc,color='r')  
plt.legend()

joblib.dump(rf,'model.save')

z=[lr\_accuracy,knn\_accuracy,dt\_accuracy,naive\_accuracy,rf\_accuracy]  
label=["LR","KNN","DT","Naive","RF"]  
A=plt.bar(label,z,width=0.5,color=['g','r','b','y','orange']) #Graphical representation of accuracies of differents Classification algorithms  
plt.xlabel('Classification Algorithms')  
plt.ylabel('Accuracy')

df.shape

clean\_df=pd.DataFrame(df)  
clean\_df

2.Flask Model:

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

import joblib

trans=joblib.load('Scalar.save')

model = joblib.load('model.save')

@app.route('/')

def helloworld():

return render\_template("base.html")

@app.route('/assesment')

def prediction():

return render\_template("index.html")

@app.route('/risk', methods = ['POST'])

def admin():

p=request.form["age"]

q= request.form["gender"]

if (q == 'f'):

q1,q2=1,0

if (q == 'm'):

q1,q2=0,1

r= request.form["housing"]

if (r == 'own'):

r1,r2,r3=0,1,0

if (r == 'free'):

r1,r2,r3=1,0,0

if (r == 'rent'):

r1,r2,r3=0,0,1

s= request.form["job"]

if (s == 'un'):

s=0

if (s == 'ur'):

s=1

if (s == 'sk'):

s=2

if (s == 'hs'):

s=3

t= request.form["saving"]

if (t == 'l'):

t1,t2,t3,t4=1,0,0,0

if (t == 'm'):

t1,t2,t3,t4=0,1,0,0

if (t == 'qr'):

t1,t2,t3,t4=0,0,1,0

if (t == 'r'):

t1,t2,t3,t4=0,0,0,1

u= request.form["checking"]

if (u == 'lt'):

u1,u2,u3=1,0,0

if (u == 'mo'):

u1,u2,u3=0,1,0

if (u == 'ri'):

u1,u2,u3=0,0,1

v= request.form["credit"]

w= request.form["duration"]

x= request.form["purpose"]

if (x == 'bu'):

x1,x2,x3,x4,x5,x6,x7,x8=1,0,0,0,0,0,0,0

if (x == 'car'):

x1,x2,x3,x4,x5,x6,x7,x8=0,1,0,0,0,0,0,0

if (x == 'da'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,1,0,0,0,0,0

if (x == 'edu'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,0,1,0,0,0,0

if (x == 'fe'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,0,0,1,0,0,0

if (x == 'rtv'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,0,0,0,1,0,0

if (x == 'rep'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,0,0,0,0,1,0

if (x == 'vo'):

x1,x2,x3,x4,x5,x6,x7,x8=0,0,0,0,0,0,0,1

y=[[int(x1),int(x2),int(x3),int(x4),int(x5),int(x6),int(x7),int(x8),int(r1),int(r2),int(r3),int(t1),int(t2),int(t3),int(t4),int(u1),int(u2),int(u3),int(q1),int(q2),int(p),int(s),int(v),int(w)]]

y=trans.transform(y)

a = model.predict(y)

if (a[0] == 0):

b ="Bad"

return render\_template("predbad.html", z = b)

else:

b ="Good"

return render\_template("predgood.html", z = b)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

1. **UI Output**

