B. Tech Program

Course: Machine Learning Lab

Course Code: DS2231

## Startup success rate classification

by

**Yash Turambekar**

(Reg. No: **219309092**)

B.Tech. 2nd Year, Section-A

Under the guidance

of

**Mr. Venkatesh Gauri Shankar**

**Assistant Professor (Sr. Scale)**

Department of Information and Technology,

School of Information Technology, Manipal University Jaipur, Jaipur

***Department of Information Technology,***

***School of IT,***

***Faculty of Engineering***

***Manipal University Jaipur, India***

**April 2023**

# Certificate

This is to certify that the project titled **“Startup success rate classification”** is a record of the bona fide work done by **Yash Turambekar** (Reg No:219309092) submitted for the partial fulfilment of the requirements for the completion of the Machine Learning Lab (DS2231) course in the **Department of Information Technology** of **Manipal University Jaipur,** during the academic session March - June 2023.

|  |
| --- |
| *Signature of the mentor* |
| Venkatesh Gauri Shankar,  Assistant Professor, (Senior Scale)  Department of IT, SCIT |

# Introduction

The aim of this project is to predict the success of startups using machine learning algorithms on available historical data. Startups are classified into acquired/closed based on data like location, type of startup, funding received etc. We apply different algorithms on the dataset and compare their accuracy.

# Methodology

# PCA

# import pandas as pd

# 

# df = pd.read\_csv('startup.csv')

# X = df.drop(['Unnamed: 0',

# 'state\_code',

# 'latitude',

# 'longitude',

# 'zip\_code',

# 'id',

# 'city',

# 'Unnamed: 6',

# 'name',

# 'labels',

# 'founded\_at',

# 'closed\_at',

# 'first\_funding\_at',

# 'last\_funding\_at',

# 'state\_code.1',

# 'category\_code',

# 'object\_id',

# 'status'],axis = 1)

# mean1 = X['age\_first\_milestone\_year'].mean()

# mean2 = X['age\_last\_milestone\_year'].mean()

# X['age\_first\_milestone\_year'].fillna(mean1, inplace = True)

# X['age\_last\_milestone\_year'].fillna(mean2, inplace = True)

# y = df['status']

# 

# from sklearn.preprocessing import LabelEncoder

# le = LabelEncoder()

# y = le.fit\_transform(y)

# 

# from sklearn.preprocessing import StandardScaler

# sc = StandardScaler()

# X = sc.fit\_transform(X)

# 

# from sklearn.decomposition import PCA

# principal = PCA(n\_components = 2)

# dfx\_pca = principal.fit(X)

# dfx\_trans = principal.transform(X)

# dfx\_trans = pd.DataFrame(data=dfx\_trans)

# 

# import matplotlib.pyplot as plt

# plt.figure(figsize=(10,6))

# plt.scatter(dfx\_trans[0],dfx\_trans[1],c=y,edgecolors='k',alpha=0.75,s=150)

# plt.grid(True)

# plt.title("Class separation using first two principal components",fontsize=20)

# plt.xlabel("Principal component-1",fontsize=15)

# plt.ylabel("Principal component-2",fontsize=15)

# plt.show()

# KNN

# *import pandas as pd*

# 

# *df = pd.read\_csv('startup.csv')*

# *X = df.drop(['Unnamed: 0',*

# *'state\_code',*

# *'latitude',*

# *'longitude',*

# *'zip\_code',*

# *'id',*

# *'city',*

# *'Unnamed: 6',*

# *'name',*

# *'labels',*

# *'founded\_at',*

# *'closed\_at',*

# *'first\_funding\_at',*

# *'last\_funding\_at',*

# *'state\_code.1',*

# *'category\_code',*

# *'object\_id',*

# *'status'],axis = 1)*

# *mean1 = X['age\_first\_milestone\_year'].mean()*

# *mean2 = X['age\_last\_milestone\_year'].mean()*

# *X['age\_first\_milestone\_year'].fillna(mean1, inplace = True)*

# *X['age\_last\_milestone\_year'].fillna(mean2, inplace = True)*

# *y= df['status']*

# 

# *from sklearn.preprocessing import LabelEncoder*

# *le = LabelEncoder()*

# *y = le.fit\_transform(y)*

# 

# *from sklearn.decomposition import PCA*

# *principal = PCA(n\_components = 8)*

# *dfx\_pca = principal.fit(X)*

# *dfx\_trans = principal.transform(X)*

# *dfx\_trans = pd.DataFrame(data=dfx\_trans)*

# 

# *from sklearn.model\_selection import train\_test\_split*

# *X\_train, X\_test, y\_train, y\_test = train\_test\_split(dfx\_trans , y, test\_size = 0.2, random\_state = 30)*

# 

# *from sklearn.preprocessing import StandardScaler*

# *sc = StandardScaler()*

# *X\_train = sc.fit\_transform(X\_train)*

# *X\_test = sc.fit\_transform(X\_test)*

# 

# *from sklearn.neighbors import KNeighborsClassifier*

# *classifier = KNeighborsClassifier(n\_neighbors = 15, metric = 'minkowski', p = 3)*

# *classifier.fit(X\_train, y\_train)*

# *y\_pred = classifier.predict(X\_test)*

# 

# *from sklearn.metrics import confusion\_matrix,accuracy\_score*

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

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

# *print("Confustion matrix:\n",cm)*

# *print("Accuracy score: ",ac)*

# Naïve Bayes

# df = pd.read\_csv('startup.csv')

# X = df.drop(['Unnamed: 0',

# 'state\_code',

# 'latitude',

# 'longitude',

# 'zip\_code',

# 'id',

# 'city',

# 'Unnamed: 6',

# 'name',

# 'labels',

# 'founded\_at',

# 'closed\_at',

# 'first\_funding\_at',

# 'last\_funding\_at',

# 'state\_code.1',

# 'category\_code',

# 'object\_id',

# 'status'],axis = 1)

# mean1 = X['age\_first\_milestone\_year'].mean()

# mean2 = X['age\_last\_milestone\_year'].mean()

# X['age\_first\_milestone\_year'].fillna(mean1, inplace = True)

# X['age\_last\_milestone\_year'].fillna(mean2, inplace = True)

# y= df['status']

# 

# from sklearn.preprocessing import LabelEncoder

# le = LabelEncoder()

# y = le.fit\_transform(y)

# 

# from sklearn.model\_selection import train\_test\_split

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X , y, test\_size = 0.2, random\_state = 30)

# 

# from sklearn.preprocessing import StandardScaler

# sc = StandardScaler()

# X\_train = sc.fit\_transform(X\_train)

# X\_test = sc.fit\_transform(X\_test)

# 

# from sklearn.naive\_bayes import GaussianNB

# classifier = GaussianNB()

# classifier.fit(X\_train, y\_train)

# y\_pred = classifier.predict(X\_test)

# 

# from sklearn.metrics import confusion\_matrix,accuracy\_score # Make confusion matrix

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

# ac = accuracy\_score(y\_test,y\_pred)

# print("Confustion matrix:\n",cm)

# print("Accuracy score: ",ac)

# Decision Tree

# import pandas as pd

# 

# df = pd.read\_csv('startup.csv')

# X = df.drop(['Unnamed: 0',

# 'state\_code',

# 'latitude',

# 'longitude',

# 'zip\_code',

# 'id',

# 'city',

# 'Unnamed: 6',

# 'name',

# 'labels',

# 'founded\_at',

# 'closed\_at',

# 'first\_funding\_at',

# 'last\_funding\_at',

# 'state\_code.1',

# 'category\_code',

# 'object\_id',

# 'status'],axis = 1)

# mean1 = X['age\_first\_milestone\_year'].mean()

# mean2 = X['age\_last\_milestone\_year'].mean()

# X['age\_first\_milestone\_year'].fillna(mean1, inplace = True)

# X['age\_last\_milestone\_year'].fillna(mean2, inplace = True)

# y= df['status']

# 

# from sklearn.preprocessing import LabelEncoder

# le = LabelEncoder()

# y = le.fit\_transform(y)

# 

# from sklearn.model\_selection import train\_test\_split

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X , y, test\_size = 0.2, random\_state = 20)

# 

# from sklearn.tree import DecisionTreeClassifier

# from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

# dtree = DecisionTreeClassifier()

# dtree.fit(X\_train,y\_train)

# y\_pred = dtree.predict(X\_test)

# print("Classification report - \n", classification\_report(y\_test,y\_pred))

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

# print("Confustion matrix: \n",cm)

# ac = accuracy\_score(y\_test, y\_pred)

# print("Accuracy score: ",ac)

# 

# import matplotlib.pyplot as plt

# import seaborn as sns

# from sklearn.tree import plot\_tree

# plt.figure(figsize=(5,5))

# sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap = 'Blues')

# plt.ylabel('Actual label')

# plt.xlabel('Predicted label')

# all\_sample\_title = 'Accuracy Score: {0}'.format(dtree.score(X\_test, y\_test))

# plt.title(all\_sample\_title, size = 15)

# plt.figure(figsize = (10,10))

# dec\_tree = plot\_tree(decision\_tree=dtree, feature\_names = X.columns ,class\_names =['0','1'] , filled = True ,precision = 4, rounded = True)

# SVM

# import pandas as pd

# 

# df = pd.read\_csv('startup.csv')

# X = df.drop(['Unnamed: 0',

# 'state\_code',

# 'latitude',

# 'longitude',

# 'zip\_code',

# 'id',

# 'city',

# 'Unnamed: 6',

# 'name',

# 'labels',

# 'founded\_at',

# 'closed\_at',

# 'first\_funding\_at',

# 'last\_funding\_at',

# 'state\_code.1',

# 'category\_code',

# 'object\_id',

# 'status'],axis = 1)

# mean1 = X['age\_first\_milestone\_year'].mean()

# mean2 = X['age\_last\_milestone\_year'].mean()

# X['age\_first\_milestone\_year'].fillna(mean1, inplace = True)

# X['age\_last\_milestone\_year'].fillna(mean2, inplace = True)

# y= df['status']

# 

# from sklearn.model\_selection import train\_test\_split

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X , y, test\_size = 0.2, random\_state = 33)

# 

# from sklearn.preprocessing import StandardScaler

# sc = StandardScaler()

# X\_train = sc.fit\_transform(X\_train)

# X\_test = sc.fit\_transform(X\_test)

# 

# from sklearn.svm import SVC

# svc = SVC(C = 1.0, random\_state = 1, kernel = 'linear')

# svc.fit(X\_train, y\_train)

# y\_pred = svc.predict(X\_test)

# 

# from sklearn.metrics import confusion\_matrix,accuracy\_score

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

# ac = accuracy\_score(y\_test,y\_pred)

# print("Confustion matrix:\n",cm)

# print("Accuracy score: ",ac)

# Results

# PCA

# 

# KNN

# 

# Naïve Bayes

# 

# Decision Tree

# 

# 

# SVM

# 

# Conclusion

# We observe that KNN, Decision Tree and SVM have good accuracy (greater than 0.8) compared to others. By feeding new test data to the model, we can predict the success/failure of startups with good accuracy.

# References

1. [https://www.w3schools.com/python/](https://www.w3schools.com/java/)
2. Geeksforgeeks.org