Prosperity Prognosticator: Machine Learning for Startup Success Prediction

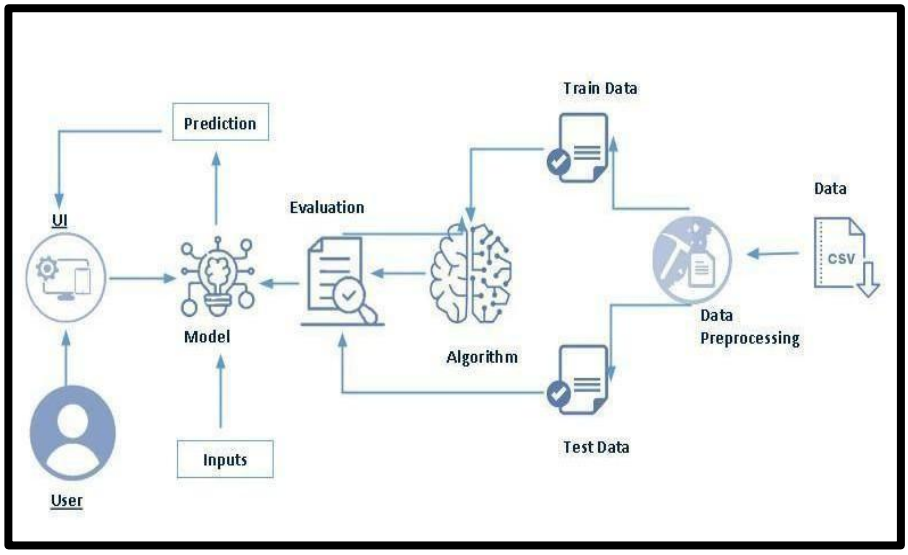
SmartInternz

www.smartinternz.com

Startup Success Prediction Using Machine Learning

Startups face high uncertainty, with many failing within their early years. Predicting startup success can help investors and founders make better decisions. This project uses machine learning to analyze key factors like funding, team, and industry to predict whether a startup is likely to succeed. The goal is to build a reliable, data-driven model for startup outcome prediction.

Technical Architecture:



Project Flow:

● User Interaction & Prediction Flow

- User interacts with the UI to enter startup-related inputs (e.g., funding amount, team size, industry, etc.).
- The input is analyzed by the machine learning model integrated into the backend.
- The model processes the input and predicts whether the startup is likely to succeed or fail.
- The prediction result is displayed on the UI for the user.

● Define Problem / Problem Understanding

- **Specify the Business Problem:**
  Predict startup success or failure using historical data and machine learning to support investment and business decisions.
- **Business Requirements:**
  Build a predictive model with a simple web interface to provide insights for founders, investors, and incubators.
- **Literature Survey:**
  Research existing academic work and solutions related to startup failure/success prediction using ML techniques.
- **Social or Business Impact:**
  Helps reduce financial risk, encourages data-driven startup decisions, and promotes startup ecosystem growth.

● Data Collection & Preparation

- **Collect the Dataset:**
  Source data from platforms like Crunchbase, Kaggle, or AngelList including features like funding, location, sector, etc.
- **Data Preparation:**
  Clean the data (remove duplicates, handle missing values), encode categorical variables, normalize/scale numerical features, and create derived features.

● Exploratory Data Analysis

- **Descriptive Statistical Analysis:**
  Analyze distributions, summary statistics, correlations between features.
- **Visual Analysis:**
  Use bar plots, boxplots, histograms, and heatmaps to visualize trends and identify key patterns.

● Model Building

- **Training the Model in Multiple Algorithms:**
  Use algorithms such as Logistic Regression, Random Forest, XGBoost, and SVM.
- **Testing the Model:**
  Use a test set or cross-validation to evaluate initial performance of each model.

● Performance Testing & Hyperparameter Tuning

- **Testing Model with Multiple Evaluation Metrics:**
  Use accuracy, precision, recall, F1-score, and ROC-AUC to assess model performance.
- **Comparing Accuracy Before & After Tuning:**
  Apply GridSearchCV or RandomizedSearchCV to tune hyperparameters and compare improvements in performance.

● Model Deployment

- **Save the Best Model:**
  Serialize the final model using joblib or pickle.
- **Integrate with Web Framework:**
  Develop a user interface using Flask or Streamlit where users input startup details and receive prediction results in real-time.

● Project Demonstration & Documentation

- **Record Explanation Video for Project End-to-End Solution:**
  Include problem statement, data analysis, model building, and deployment steps in the demo.

- **Project Documentation – Step-by-Step Project Development Procedure:**
  Document each phase clearly with code, screenshots, and explanations for reproducibility and understanding.

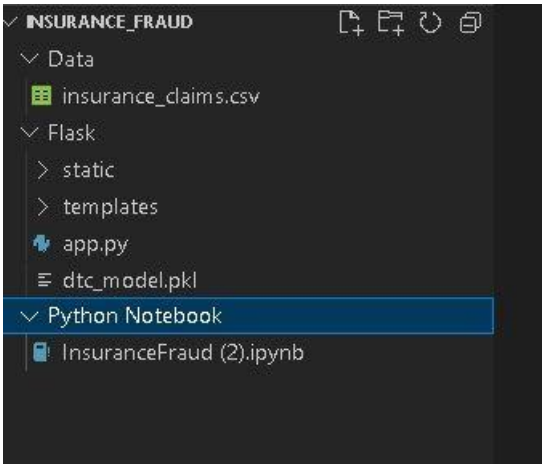
Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Dtc_model.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedure for building the model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Startups face a high failure rate, with many failing within the first few years due to poor decision-making, lack of market demand, or insufficient resources. Predicting the likelihood of a startup's success using machine learning can provide investors, founders, and stakeholders with valuable insights. This project aims to build a model that analyzes historical startup data to predict the probability of success, helping reduce investment risks and improving decision-making.

Activity 2: Business requirements

A startup success prediction project may involve the following business requirements:

- **High Prediction Accuracy**: The model should accurately classify startups as likely to succeed or fail based on relevant features such as funding amount, team background, market sector, etc.
- **Scalability**: The system should be able to handle large datasets and new entries as more startup data becomes available.
- **Interpretability**: The model should provide understandable insights, allowing users to understand which features contribute to predictions.
- **Compliance**: The system must ensure data privacy and adhere to regulations regarding financial and personal data usage.
- **User Accessibility**: The prediction tool should be accessible via a user-friendly interface for investors, analysts, and entrepreneurs.

Activity 3: Literature Survey (Student Will Write)

A literature survey for startup success prediction would involve reviewing research papers, case studies, and industry reports that have analyzed startup success factors. It would explore various machine learning algorithms used in previous studies, the features considered most impactful, and the datasets utilized. The survey will also identify limitations in existing models and highlight how the current project aims to address them. This review will guide model selection, feature engineering, and evaluation criteria.

Activity 4: Social or Business Impact.

 Social Impact:
Encouraging Innovation – By reducing the uncertainty around startup viability, more individuals may be encouraged to pursue entrepreneurial ventures, fostering innovation and job creation.
 Business Model/Impact:
Informed Investment Decisions – Investors can use the prediction model to evaluate startups more effectively, optimizing their portfolios and minimizing financial risk. Additionally, incubators and accelerators can use the system to select promising startups for funding and support.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:
https://www.kaggle.com/api/v1/datasets/download/manishkc06/startup-success-prediction?dataset_version_number=1

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```
#IMPORTING THE LIBRARIES
import numpy as np #Linear algebra
import pandas as pd #data processing
pd.set_option('Display.max_column',None)
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report,confusion_matrix,accur
from sklearn.preprocessing import LabelEncoder,StandardScaler
import pickle
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.preprocessing import FunctionTransformer
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
df = pd.read_csv(data)
df.head()
```

	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed: 6	name	labels	...	object_id	has_VC	has_angel	has_roundA
0	1005	CA	42.358880	-71.056820	92101	c:6669	San Diego	NaN	Bandsintown	1	...	c:6669	0	1	0
1	204	CA	37.238916	-121.973718	95032	c:16283	Los Gatos	NaN	TriCipher	1	...	c:16283	1	0	0
2	1001	CA	32.901049	-117.192656	92121	c:65620	San Diego	San Diego CA 92121	Plixi	1	...	c:65620	0	0	1
3	738	CA	37.320309	-122.050040	95014	c:42668	Cupertino	Cupertino CA 95014	Solidcore Systems	1	...	c:42668	0	0	0
4	1002	CA	37.779281	-122.419236	94105	c:65806	San Francisco	San Francisco CA 94105	Inhale Digital	0	...	c:65806	1	1	0

5 rows × 49 columns

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

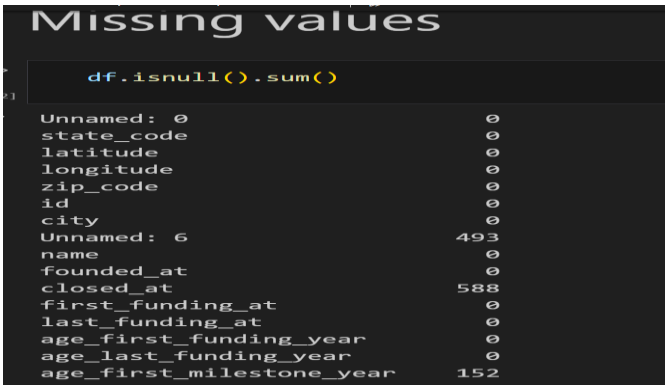
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

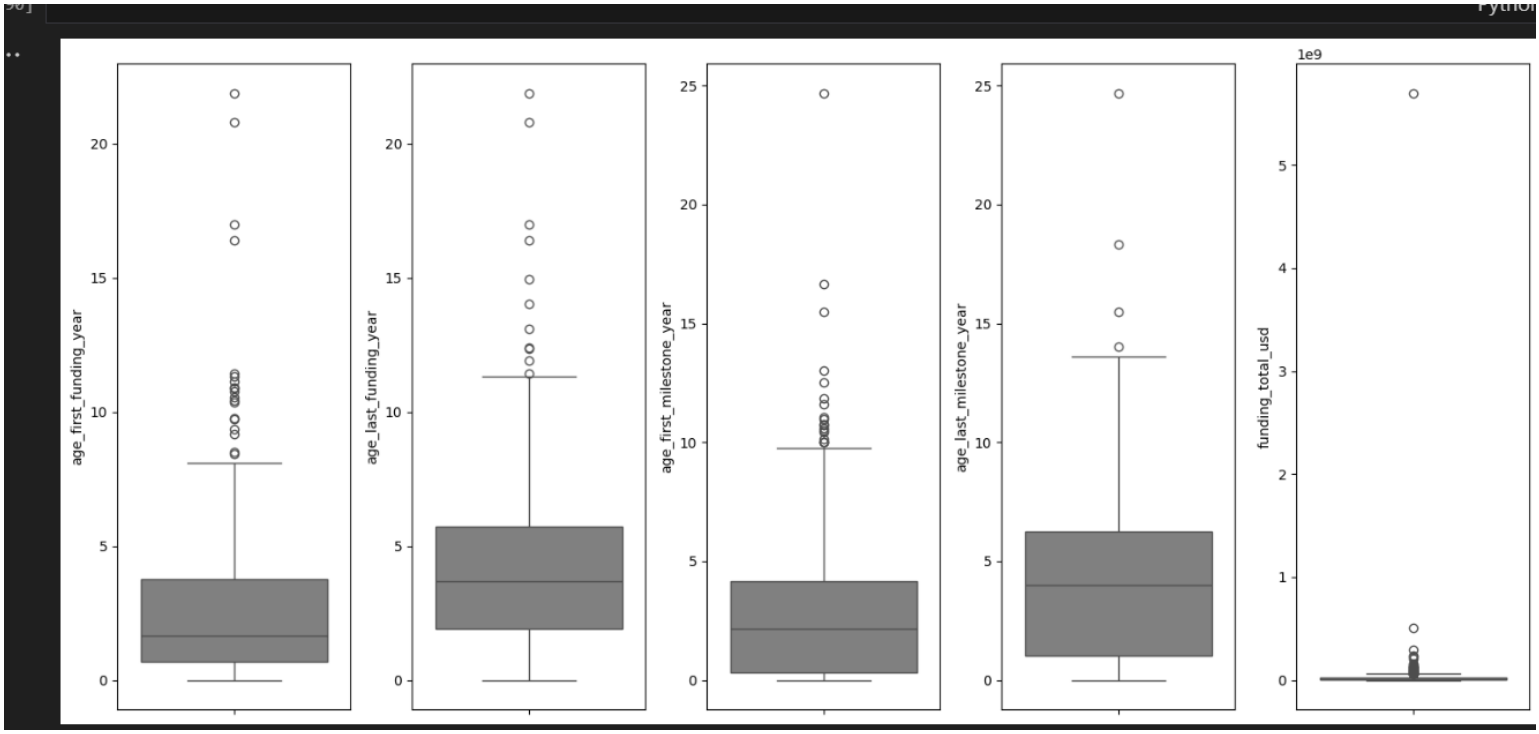
- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



Activity 2.2: Handling Outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of funding and milestone year feature with some mathematical formula.

- From the below diagram, we could visualize that the feature has outliers. Boxplot from seaborn library is used here.

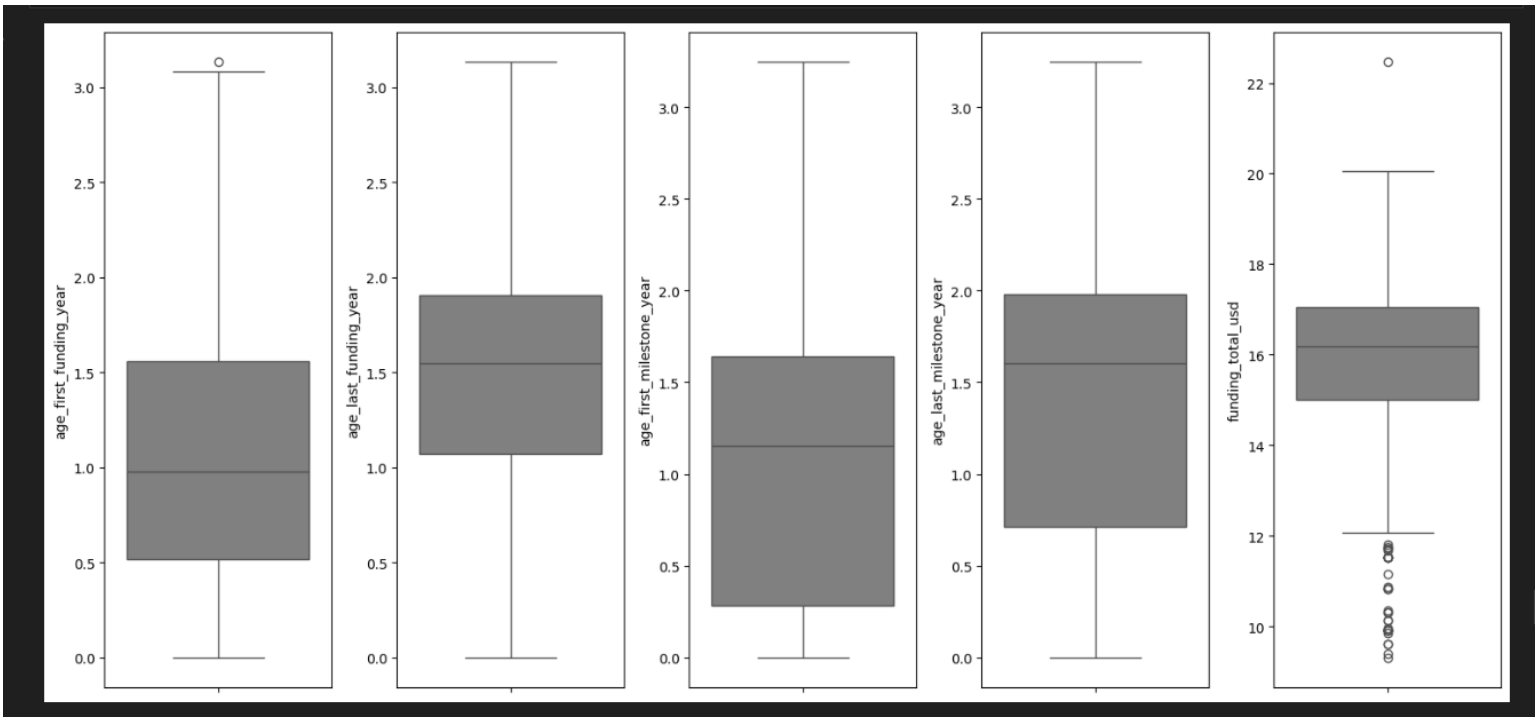



```
df["age_first_funding_year"] = np.log1p(df["age_first_funding_year"])
df["age_last_funding_year"] = np.log1p(df["age_last_funding_year"])
df["age_first_milestone_year"] = np.log1p(df["age_first_milestone_year"])
df["age_last_milestone_year"] = np.log1p(df["age_last_milestone_year"])
df["funding_total_usd"] = np.log1p(df["funding_total_usd"])
```

Python

```
features = ['age_first_funding_year','age_last_funding_year','age_first_milestone_ye

plt.figure(figsize=(15, 7))
for i in range(0, len(features)):
    plt.subplot(1, len(features), i+1)
    sns.boxplot(y=df[features[i]], color='gray', orient='v')
    plt.tight_layout()
```


age_first_funding_year
age_last_funding_year
age_first_milestone_year
age_last_milestone_year
funding_total_usd

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	Unnamed: 0	latitude	longitude	labels	age_first_funding_year	age_last_funding_year	age_first_milestone_year	age_last_milestone_year
count	923.000000	923.000000	923.000000	923.000000	923.000000	923.000000	771.000000	771.000000
mean	572.297941	38.517442	-103.539212	0.646804	2.235630	3.931456	3.055353	4.754423
std	333.585431	3.741497	22.394167	0.478222	2.510449	2.967910	2.977057	3.212107
min	1.000000	25.752358	-122.756956	0.000000	-9.046600	-9.046600	-14.169900	-7.005500
25%	283.500000	37.388869	-122.198732	0.000000	0.576700	1.669850	1.000000	2.411000
50%	577.000000	37.779281	-118.374037	1.000000	1.446600	3.528800	2.520500	4.476700
75%	866.500000	40.730646	-77.214731	1.000000	3.575350	5.560250	4.686300	6.753400
max	1153.000000	59.335232	18.057121	1.000000	21.895900	21.895900	24.684900	24.684900

8 rows × 35 columns

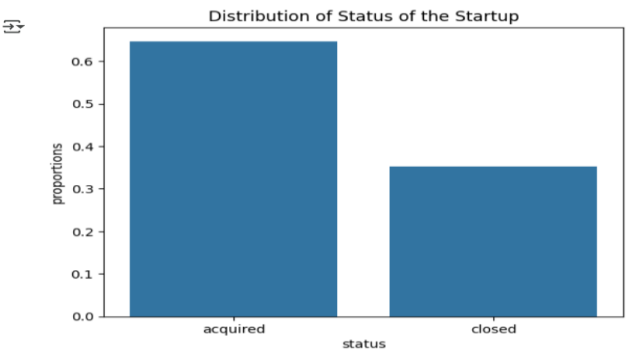
Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

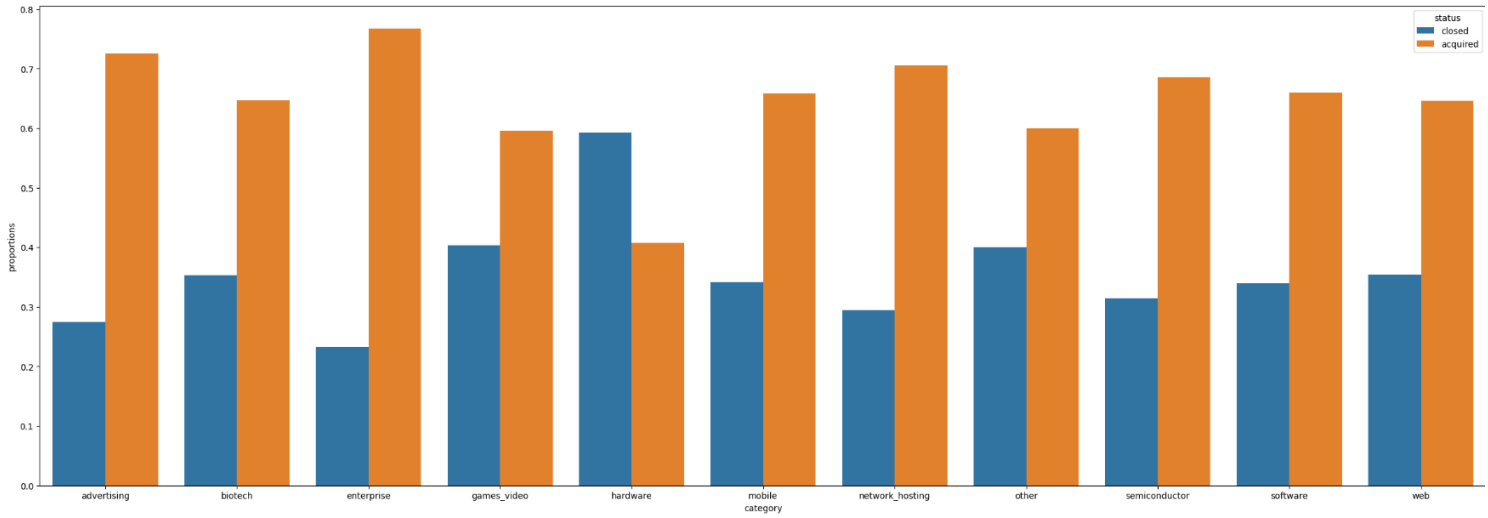
Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Piechart and countplot.

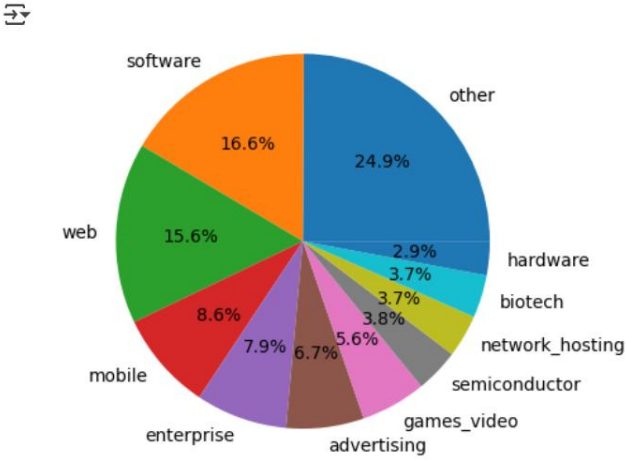
Seaborn package provides a wonderful function countplot. It is more useful for categorical features. With the help of countplot, we can Number of unique values in the feature.



```
<Axes: xlabel='category', ylabel='proportions'>
```



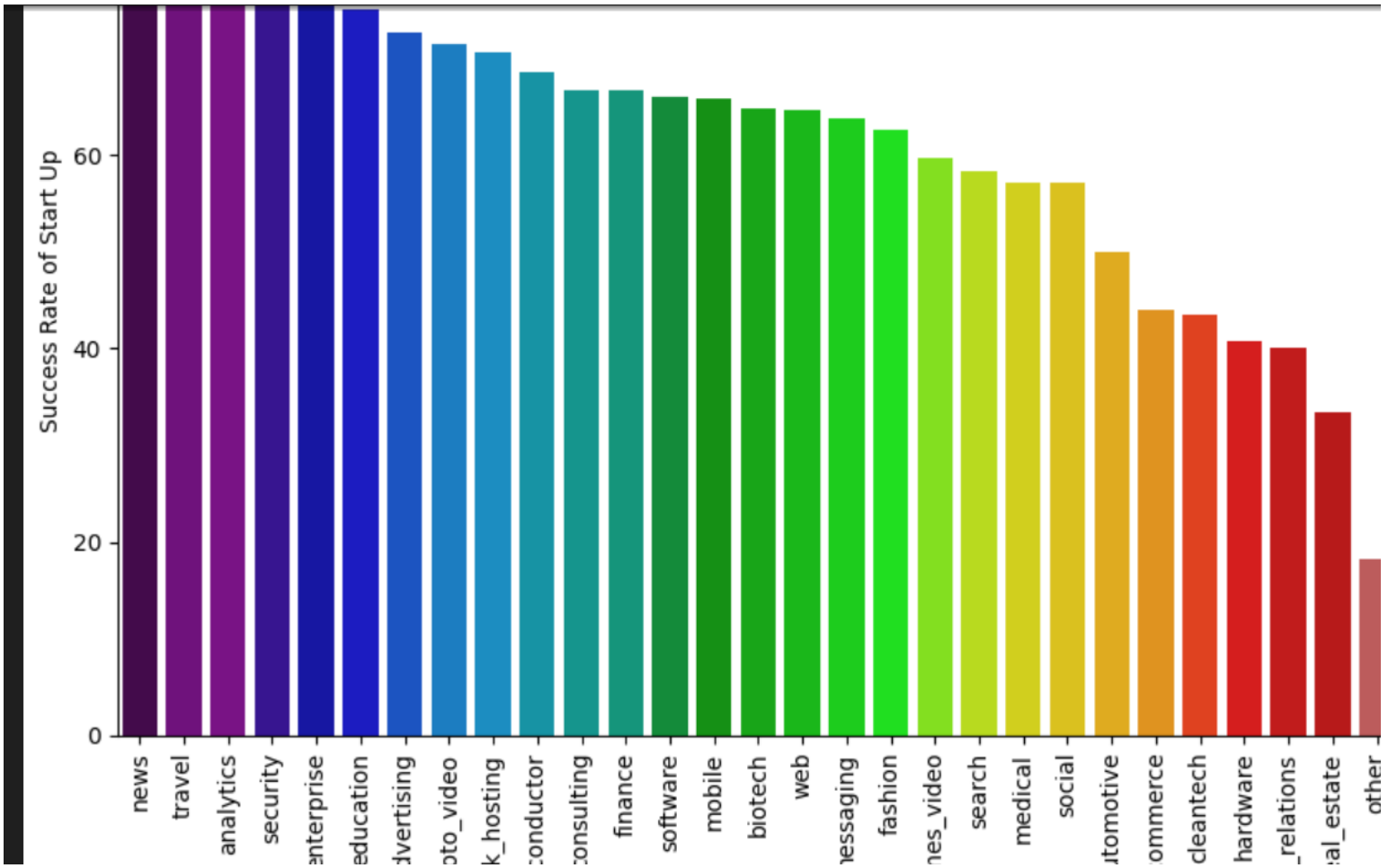
- The above chart describes the probability of success of a startup bases on different categories of startup.

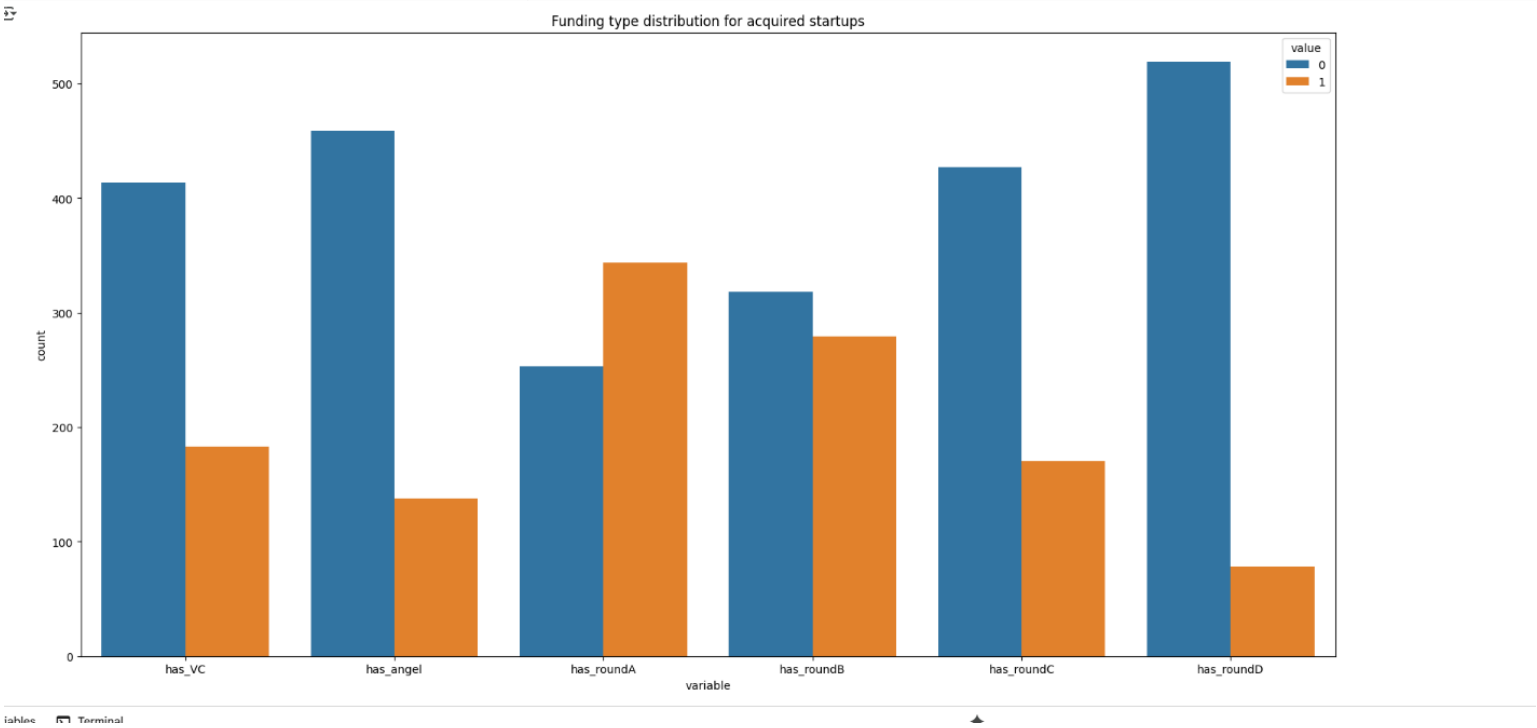


Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we can used barplot.

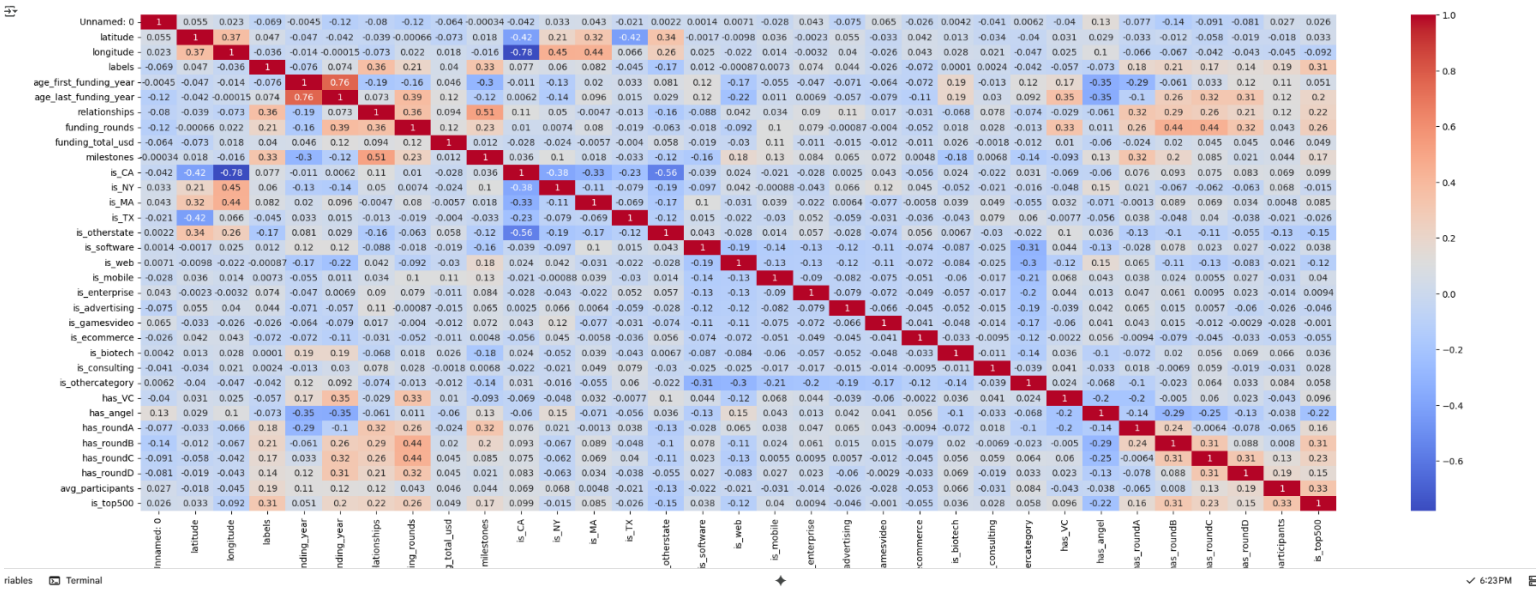
- Barplot is used here. As a 1st parameter we are passing success rate and as a 2nd parameter we are passing category
- From the below plot you can understand that distribution of category of startup and success rate





Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.



Encoding the Categorical Features:

- The categorical Features are can't be passed directly to the Machine Learning Model. So we convert them into Numerical data based on their order. This Technique is called Encoding.
- Here we are importing Label Encoder from the Sklearn Library.
- Here we are applying fit_transform to transform the categorical features to numerical features.

```
[36] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     for i in daata1.columns:
         if daata1[i].dtype=='0':
           daata1[i] = le.fit_transform(daata1[i])
```

Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
[103]  X = df.drop('status', axis = 1) #
       y = df['status']

[104]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
```

Generate + Code + Markdown

Scaling

- Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.
- Here we are using Standard Scaler.
- This scales the data to have a mean of 0 and a standard deviation of 1. The formula is given by:
  X_scaled = (X - X_mean) / X_std

```
from sklearn.preprocessing import StandardScaler
Std_scaler=StandardScaler()
```

```
X_train = Std_scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train, columns=x.columns)
```

```
X_test = Std_scaler.transform(X_test)
X_test = pd.DataFrame(X_test, columns=x.columns)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(X_train,y_train)
y_pred=dtc.predict(X_test)
dtc_train_acc=accuracy_score(y_train,dtc.predict(X_train))
dtc_test_acc=accuracy_score(y_test,y_pred)
```

Activity 1.2: Random forest model

First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(criterion='entropy',max_depth=10,max_features='sqrt',min_samples_leaf= 1, min_samples_split= 3, n_estimators= 140)
rfc.fit(X_train,y_train)
y_pred=rfc.predict(X_test)
rfc_train_acc=100*accuracy_score(y_train,rfc.predict(X_train))
rfc_test_acc=100*accuracy_score(y_test,y_pred)
```

Activity 1.3: KNN model

KNN Model is imported from sklearn Library then KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=30)
knn.fit(X_train,y_train)
y_pred=knn.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Activity 1.4: Logistic Regression model

Logistic Regression Model is imported from sklearn Library then Logistic Regression algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix is done.

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, classification_report
lg = LogisticRegressionCV(solver='lbfgs', max_iter=5000, cv=10)
lg.fit(X_train, y_train)
print(confusion_matrix(y_test,lrg_pred))
```

Activity 1.5: Naïve Bayes model

Naïve Bayes Model is imported from sklearn Library then Naïve Bayes algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. We can find the Train and Test accuracy by X_train and X_test.

```
from sklearn.naive_bayes import CategoricalNB,GaussianNB
gnb=GaussianNB()
model_2=gnb.fit(X_train,y_train)
predict_log=model_2.predict(X_test)
print("Training Accuracy",100*accuracy_score(model_2.predict(X_train),y_train))
print("Testing Accuracy",100*accuracy_score(y_test,predict_log))
```

Activity 1.6: SVM model

SVM Model is imported from sklearn Library then SVM algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
: from sklearn.svm import SVC

  svc = SVC()
  svc.fit(X_train, y_train)
  y_pred = svc.predict(X_test)
```

```
: svc_train_acc = accuracy_score(y_train, svc.predict(X_train))
  svc_test_acc = accuracy_score(y_test, y_pred)
  print(f"Training accuracy of SVC : {svc_train_acc}")
  print(f"Test accuracy of SVC : {svc_test_acc}")
  print(confusion_matrix(y_test,y_pred))
  print(classification_report(y_test,y_pred))
```

Activity 2: Testing the model

Here we have tested with Decision Tree algorithm. You can test with all algorithm. With the help of predict() function.

```
b_dtc.predict([[328,521585,2012,12,250,1000,1406.91,5600,1,100,25,25,50000,0,120,23,56,52,1,123,2,3,1,0,2,1,150000,2,25,2002]])
```

```
]: array([0])
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above four models, the compareModel function is defined.

```
def comparison(X_test,y_test):
    print("logistic Regression: ",100*accuracy_score(y_test, lrg_pred))
    print("-"*100)
    print("KNN",100*accuracy_score(y_test,predict_log))
    print("-"*100)
    print("SVM",100*svc_train_acc)
    print("-"*100)
    print("Naive-Bayes",100*accuracy_score(model_2.predict(X_test),y_test))
    print("-"*100)
    print("Decision Tree",100*b_dtc_test_acc)
    print("-"*100)
    print("Random Forest",100*b_rfc_test_acc)
    print("-"*100)
```

```
comparison(X_test,y_test)
```

```
logistic Regression:  66.5
----------------------------------------------------------------------------------------------------
KNN 66.5
----------------------------------------------------------------------------------------------------
SVM 95.57377049180327
----------------------------------------------------------------------------------------------------
Naive-Bayes 66.5
----------------------------------------------------------------------------------------------------
Decision Tree 85.0
----------------------------------------------------------------------------------------------------
Random Forest 80.0
----------------------------------------------------------------------------------------------------
```

```
# Logistic Regression
print(confusion_matrix(y_test,lrg_pred))
print(classification_report(y_test,lrg_pred))
```

```
[[91 52]
 [15 42]]
              precision    recall  f1-score   support

           0       0.86      0.64      0.73       143
           1       0.45      0.74      0.56        57

    accuracy                           0.67       200
   macro avg       0.65      0.69      0.64       200
weighted avg       0.74      0.67      0.68       200
```

```
# SVM
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[101  42]
 [ 18  39]]
              precision    recall  f1-score   support

           0       0.85      0.71      0.77       143
           1       0.48      0.68      0.57        57

    accuracy                           0.70       200
   macro avg       0.67      0.70      0.67       200
weighted avg       0.74      0.70      0.71       200
```

```
# Decision Tree
print("Confusion Matrix \n",confusion_matrix(y_test,y_pred),"\n")
print("classification_report \n",classification_report(y_test,y_pred))
```

```
Confusion Matrix
 [[101  42]
 [ 18  39]]

classification_report
              precision    recall  f1-score   support

           0       0.85      0.71      0.77       143
           1       0.48      0.68      0.57        57

    accuracy                           0.70       200
   macro avg       0.67      0.70      0.67       200
weighted avg       0.74      0.70      0.71       200
```

After calling the function, the results of models are displayed as output. From the above models Decision Tree is performing well.

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional. For this project it is not required.)

Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well.

Note: To understand cross validation, refer to this link

```
from sklearn.model_selection import cross_val_score
cv=cross_val_score(b_dtc,X_train,y_train,cv=11)
print(cv)
```

```
[0.8018018  0.82882883 0.86486486 0.85585586 0.90990991 0.89189189
 0.81981982 0.83783784 0.87387387 0.88288288 0.91818182]
```

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance.This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
filename='dtc_model.pkl'
pickle.dump(b_dtc, open(filename,'wb'))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Page:

For this project create HTML file namely

- index.html

and save them in the templates folder. Refer this link for templates.

Activity 2.2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

```
app = Flask(__name__)

#Load the model
model = pickle.load(open("C:/Users/Sohan/OneDrive/Desktop/python/Ma

#List of exactly 36 features
features = [
    'age_first_funding_year', 'age_last_funding_year', 'age_first_m
    'funding_rounds', 'funding_total_usd', 'milestones', 'is_CA', '
    'is_software', 'is_web', 'is_mobile', 'is_enterprise', 'is_adve
    'is_biotech', 'is_consulting', 'is_othercategory', 'has_VC', 'h
    'has_roundC', 'has_roundD', 'avg_participants', 'is_top500', 'h
    'invalid_startup', 'age_startup_year', 'tier_relationship'
]
```

Render HTML page:

```
# ✅ Homepage route
@app.route('/')
def home():
    return render_template("index.html")  # This should be

# ✅ Prediction form route
@app.route('/predict', methods=['GET', 'POST'])
def predict():
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

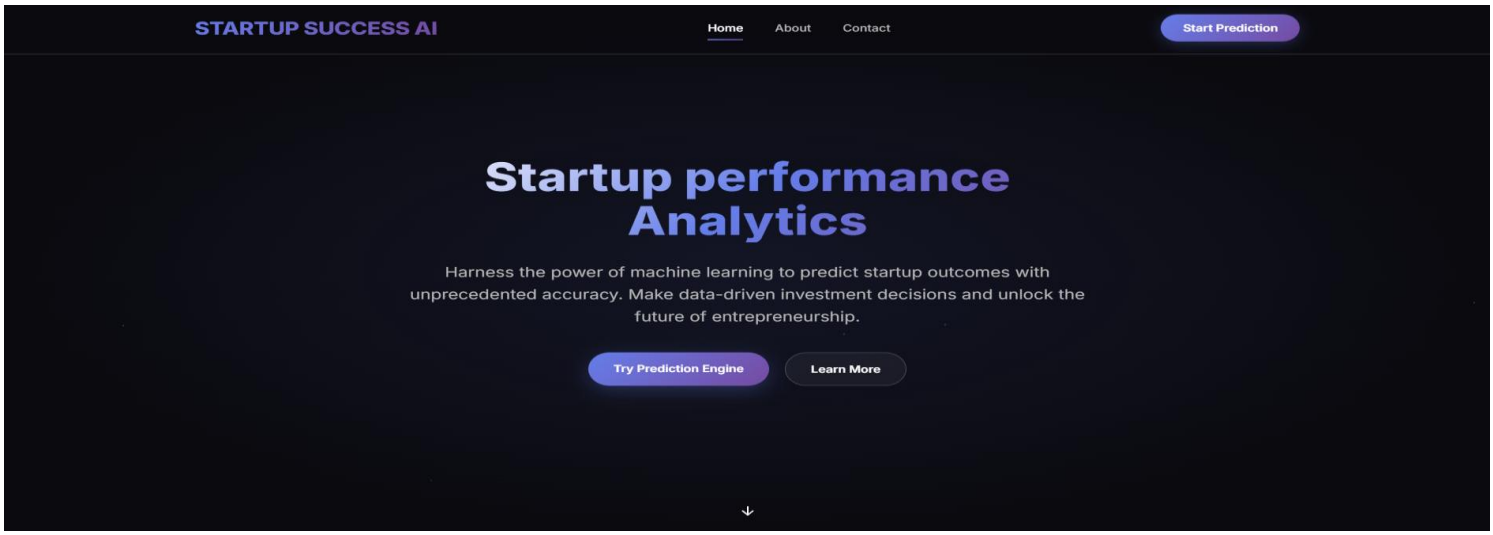
```
if __name__ == '__main__':
    app.run(debug=True)
```

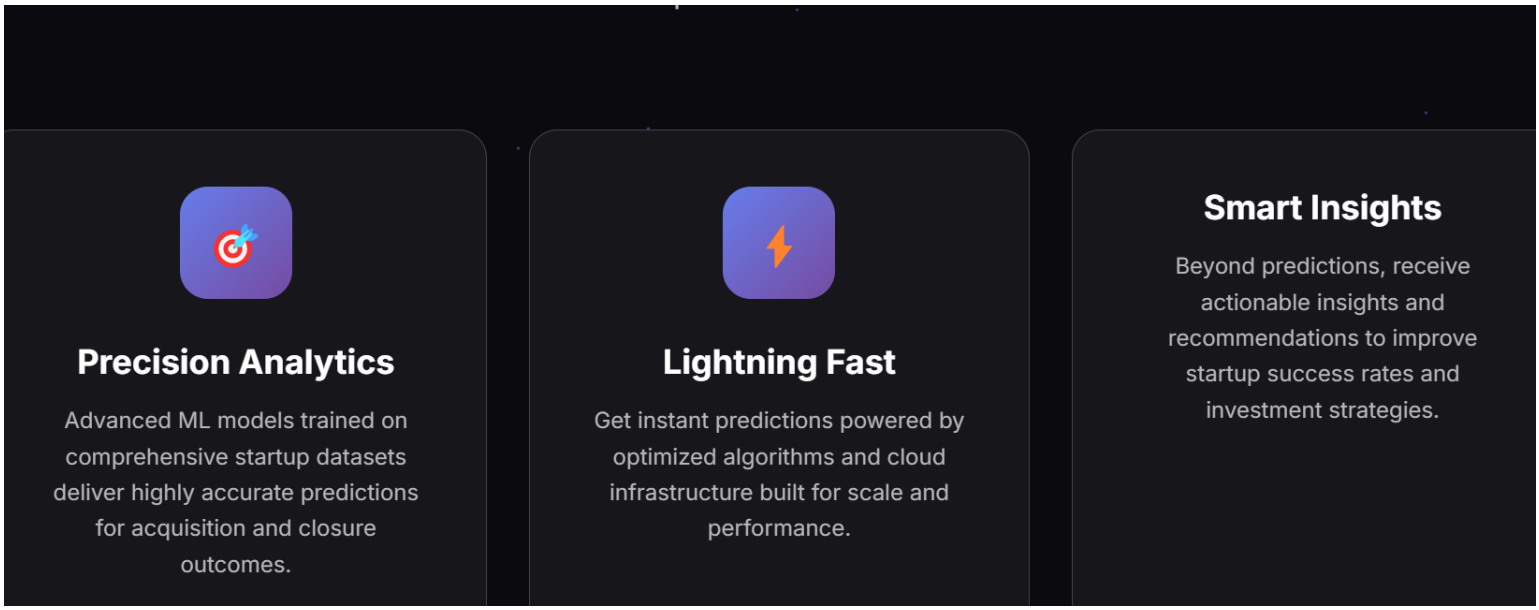
Activity 2.3: Run the web application

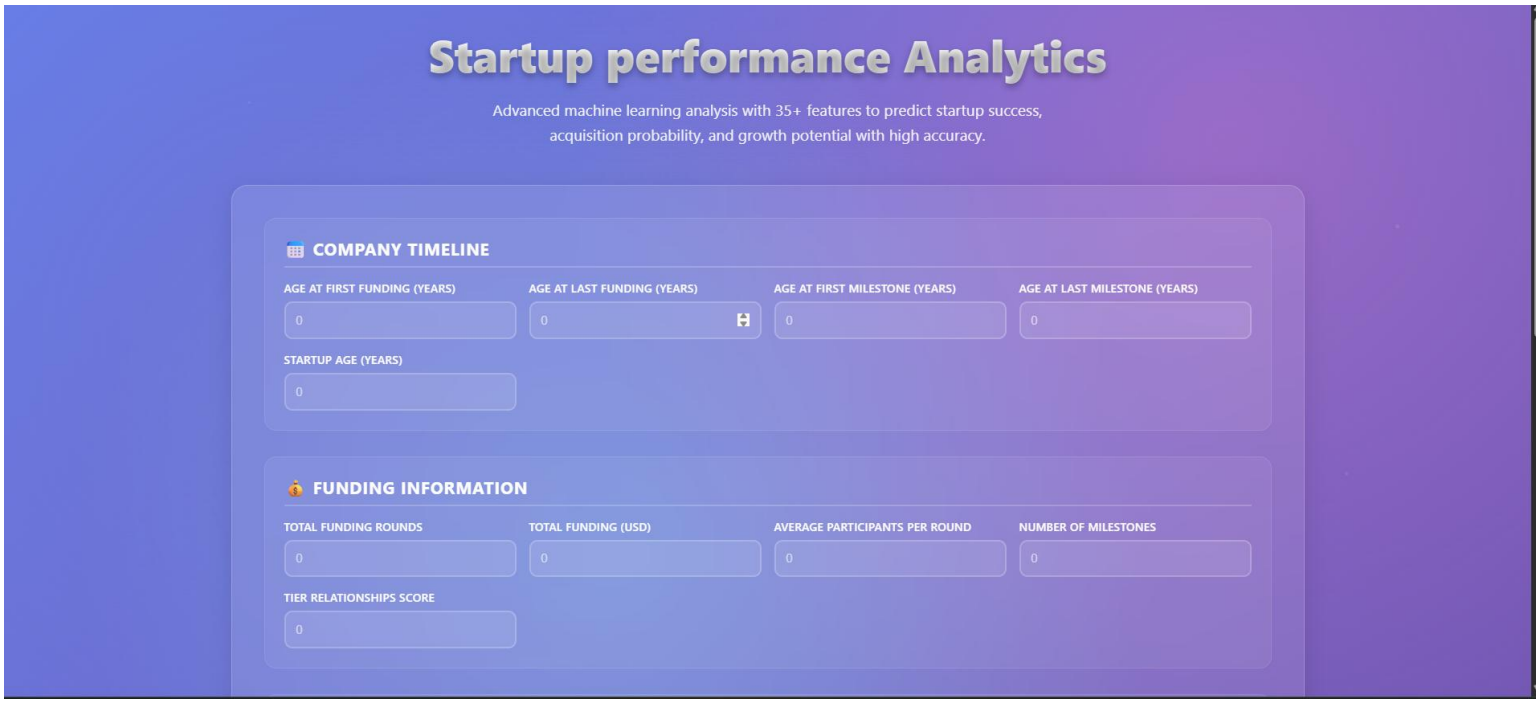
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

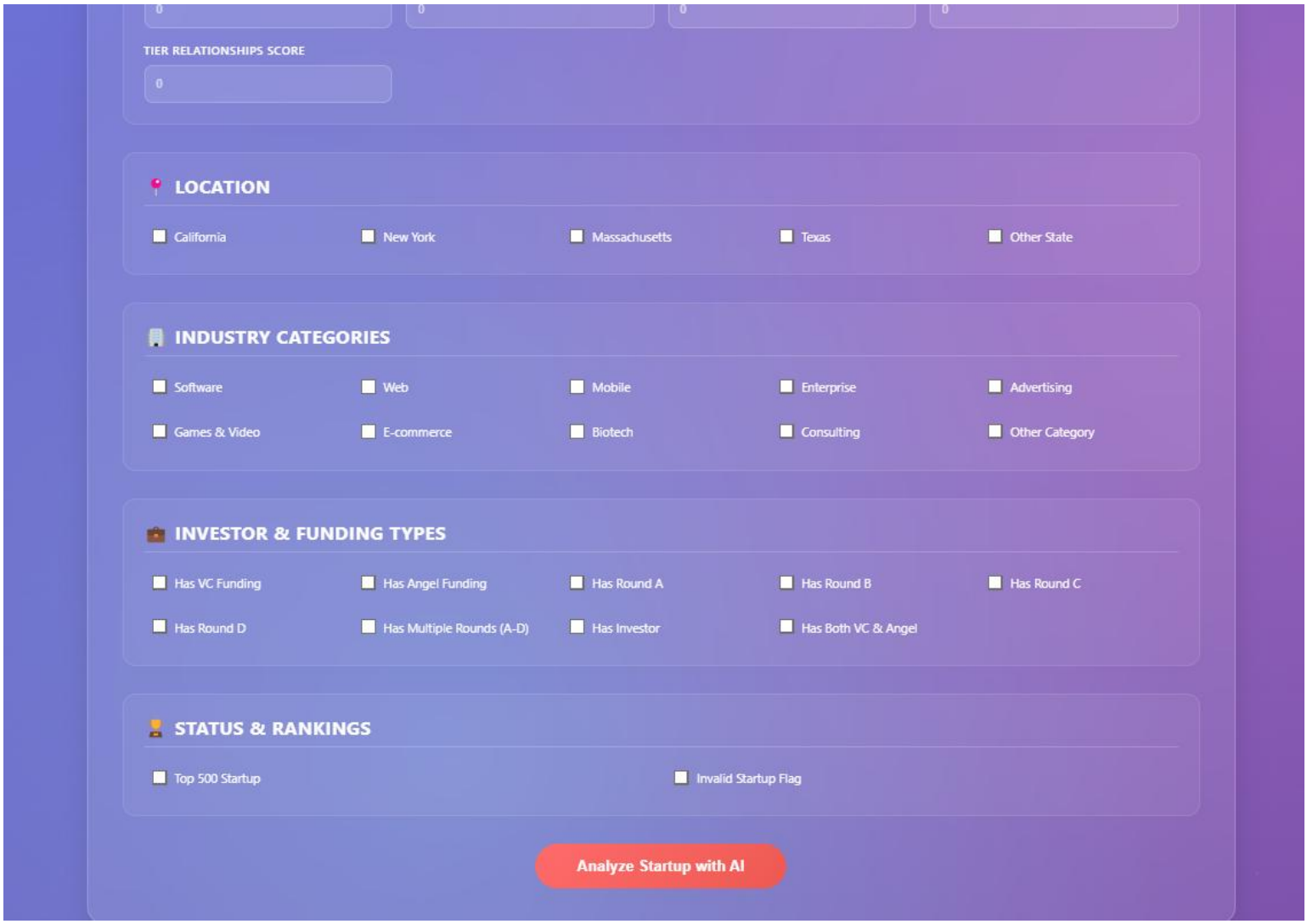
```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a p
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result




Precision Analytics
Advanced ML models trained on comprehensive startup datasets deliver highly accurate predictions for acquisition and closure outcomes.
Lightning Fast
Get instant predictions powered by optimized algorithms and cloud infrastructure built for scale and performance.
Smart Insights
Beyond predictions, receive actionable insights and recommendations to improve startup success rates and investment strategies.


Startup performance Analytics
Advanced machine learning analysis with 35+ features to predict startup success, acquisition probability, and growth potential with high accuracy.
COMPANY TIMELINE
AGE AT FIRST FUNDING (YEARS)
AGE AT LAST FUNDING (YEARS)
AGE AT FIRST MILESTONE (YEARS)
AGE AT LAST MILESTONE (YEARS)
STARTUP AGE (YEARS)
FUNDING INFORMATION
TOTAL FUNDING ROUNDS
TOTAL FUNDING (USD)
AVERAGE PARTICIPANTS PER ROUND
NUMBER OF MILESTONES
TIER RELATIONSHIPS SCORE


0
0
0
0
TIER RELATIONSHIPS SCORE
0
LOCATION
California
New York
Massachusetts
Texas
Other State
INDUSTRY CATEGORIES
Software
Web
Mobile
Enterprise
Advertising
Games & Video
E-commerce
Biotech
Consulting
Other Category
INVESTOR & FUNDING TYPES
Has VC Funding
Has Angel Funding
Has Round A
Has Round B
Has Round C
Has Round D
Has Multiple Rounds (A-D)
Has Investor
Has Both VC & Angel
STATUS & RANKINGS
Top 500 Startup
Invalid Startup Flag
Analyze Startup with AI

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided