PROSPERITY PROGNOSTICATOR:
MACHINE LEARNING STARTUP SUCCESS PREDICTION

TABLE OF CONTENTS:-

I. INTRODUCTION

1.1 OVERVIEW

Startups have recently gained a lot of interest on a global scale. The number of new startups throughout the world has significantly surpassed those from the previous year as a consequence of the worldwide pandemic, which has caused a real startup boom. This surge in entrepreneurship is attributed to workers who were laid off and started their own businesses. Startups are widely seen as important drivers of economic expansion and job creation. Startups are now harmful to societal improvement. Through innovation and scalable technology, they may produce significant solutions and hence serve as engines for socio-economic development and transformation. People now have a new source of income thanks to startups. One in five microbusinesses established during the COVID time period were started. by people who were classified as unemployed. 12% was the pre-COVID This demonstrates how a lot of individuals are able to break out from the cycle of poverty thanks to microbusinesses and startups. Since they increase the economy's competitiveness and dynamism, startups are the most dynamic economic entities now operating. As a result, the economy continues to be robust, active, and industrious, and it becomes more difficult for individual businesses to rest on their laurels. However, not every business that was established in recent years was successful. This presents a fairly bleak picture for future business owners. Such failures can be attributed to a wide range of factors. Lack of market demand is one of the causes. Many ventures Centre on a product whose demand has either faded over time or has never been. The absence of proper financial means might also be a factor. Many startups simply run out of the resources they need to continue operating. Unsuitable managerial behaviour might be another factor decisions, acts of God such as Covid, or even strong competition. This machine learning models include Logistic Regression, Decision Trees, Random Forest. The information obtained for the same mostly focuses on monetary data, such as valuations and the funds raised in each round of venture capital. These models may be used not only by different startup stakeholders to evaluate their development but also by potential venture capitalists wanting to invest in the business. Since they are the greatest employers of workers, small and medium-sized businesses (SMEs) are regarded as one of the foundational elements of the nation

1.2 PURPOSE:

This project uses machine learning to predict the success of startup companies based on various features such as funding, location, industry, and team size. The goal is to help investors and entrepreneurs make more informed decisions about which startups to invest in or launch or the startup will be successful or not.

2.LITERATURE SURVEY

2.1EXISTING PROBLEM

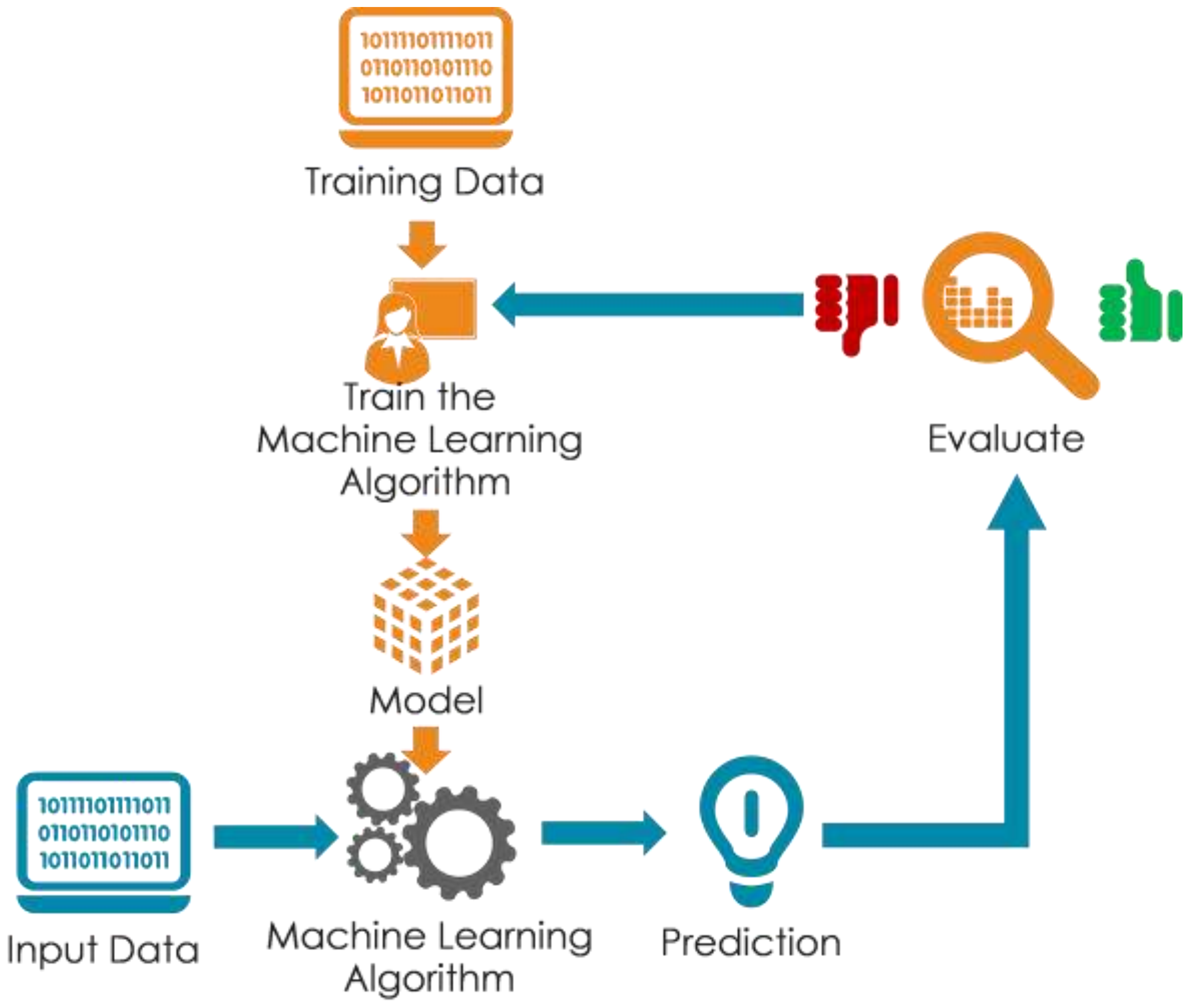
- . The prediction model of startup success is an important tool for the business vendors to make decision according to factors. Startup performance prediction is difficult task due to an analysis of various relationships among data. Researches also incorporates the social media dataset to increase the performance of the prediction. Various models were applied in the different data to increase the prediction performance and analysis of important factors.
- The prediction model is implemented based on machine learning techniques and strategic analysis. The data is processed in such a way that the probability of success or failure is calculated in the pre-start-up phase.

2.2 PROPOSED SOLUTION

Our main goal was to determine whether an already-running startup will prosper or fail. In order to make a reliable prediction, we employed a data set for predicting startup success that comprised both quantitative and categorical variables. By merging machine learning methods with the Python programming language, we aimed to do this. In order to determine whether or not a startup will prosper, we applied machine learning algorithms. This project's objective was to offer various classification-based insights. For this research, we used three algorithms: Gradient Boosting, Adaptive Boosting, Random Forest, K Nearest Neighbor (KNN), and Logistic Regression. We developed a model that can predict the success of a business in order to assess the startup behavior based on many characteristics and discover which variables have the most impact on startup success. By developing and contrasting various machine learning models, we were able to determine the most effective method for predicting a startup's success rate. Since our output is binary, we chose five machine learning methods for categorization and improved them to produce better results. The algorithm with the highest accuracy was determined to be the best algorithm for our problem dataset after analyzing the results of all machine learning algorithms. The algorithms' outputs were compared, and boosting algorithms had a greater accuracy. The boosting techniques, adaptive boosting and gradient boosting, provide a better indication of a startup's likelihood of success.

3.THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM


Training Data
Train the Machine Learning Algorithm
Evaluate
Model
Input Data
Machine Learning Algorithm
Prediction

3.2 SOFTWARE DESIGNING

The following is the Software required to complete this project:

➢ **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

➢ **Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical startup data, longitude, latitude, and other relevant features.

➢ **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

➢ **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn for custom Python code to enhance the model's efficiency.

➢ **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered.

➢ **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict startup success based on historical data.

➢ **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view startup success predictions, location, and recommended information.

➢ Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality o

EXPERIMENTAL INVESTIGATION

State_code: code where State the startup is located.

Latitude: Startup success prediction using key factors.

Longitude: Predictive insights through data analysis and modeling.

Zip_code: Zip code of the startup's location.

ID: Unique identifier for the startup.

City: City where the startup is located.

Name: Name of the startup.

Founded_at: Date when the startup was founded.

Closed_at: Date when the startup ceased operations or closed down.

Age_First_Funding_Year: The number of years from a startup's founding to its first funding.

Age Last Funding Year: The number of years from a startup's founding to its most recent funding.

Age First Milestone Year: The number of years from a startup's founding to its first significant milestone.

Age Last Milestone Year: The number of years from a startup's founding to its most recent significant milestone.

Relationships: The number of strategic partnerships and connections a startup has.

Funding Rounds: The total number of investment rounds a startup has secured.

Funding Total USD: The total amount of funding a startup has received in US dollars.

Milestones: Significant achievements or events in a startup's development timeline.

Is CA: Indicator of whether a startup is based in California

Is NY: Indicator of whether a startup is based in New York.

Is MA: Indicator of whether a startup is based in Massachusetts.

Is TX: Indicator of whether a startup is based in Texas.

Is Other State: Indicator of whether a startup is based in a state other than NY, MA, or TX.

Is Software: Indicator of whether a startup operates in the software industry.

Is Mobile: Indicator of whether a startup operates in the mobile app development industry.

Is Enterprise: Indicator of whether a startup operates in the enterprise software sector.

Is Advertising: Indicator of whether a startup operates in the advertising industry.

Is Games Video: Indicator of whether a startup operates in the video games industry.

Is Ecommerce: Indicator of whether a startup operates in the ecommerce sector.

Is Biotech: Indicator of whether a startup operates in the biotechnology industry.

Is Consulting: Indicator of whether a startup operates in the consulting sector.

Is Other Category: Indicator of whether a startup operates in a category other than software, mobile, enterprise, advertising, games video, ecommerce, biotech, or consulting.

Has VC: Indicator of whether a startup has received venture capital funding.

Has Angel: Indicator of whether a startup has received angel investment.

Has Round A: Indicator of whether a startup has completed Series A funding round.

Has Round B: Indicator of whether a startup has completed Series B funding round.

Has Round C: Indicator of whether a startup has completed Series C funding round.

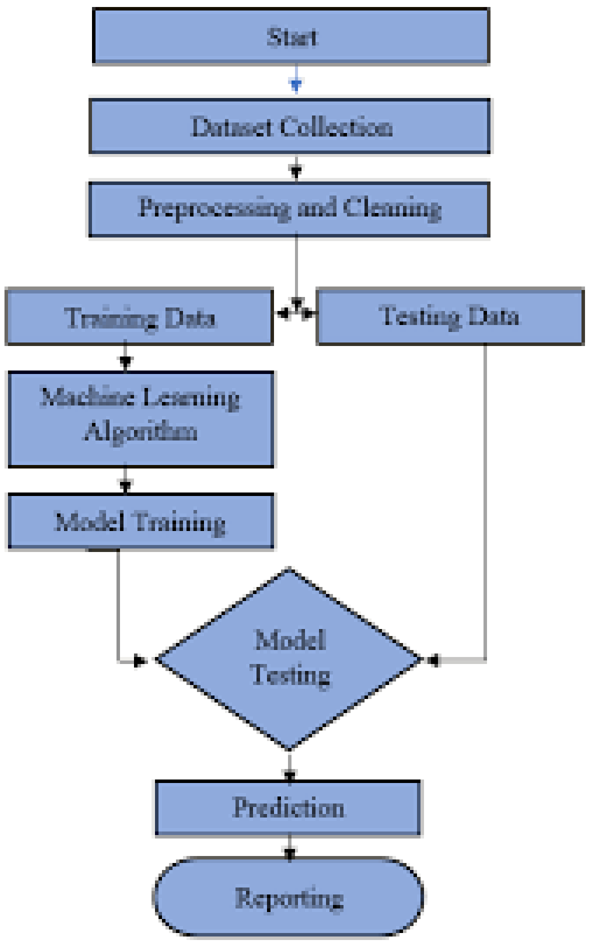
Has Round D: Indicator of whether a startup has completed Series D funding round

Avg Participants: Average number of participants in a startup's funding rounds.

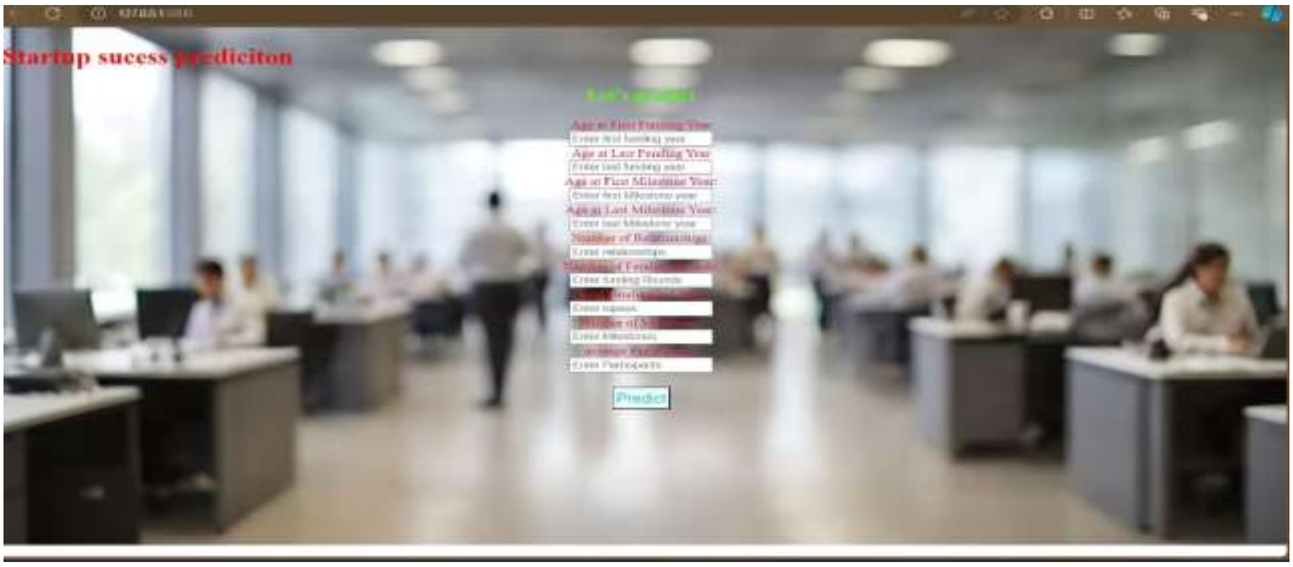
Is Top500: Indicator of whether a startup is ranked among the top 500 companies in its industry.

Status: Current status or stage of development of the startup.

5.FLOWCHART


Start
Dataset Collection
Preprocessing and Cleaning
Training Data
Testing Data
Machine Learning Algorithm
Model Training
Model Testing
Prediction
Reporting

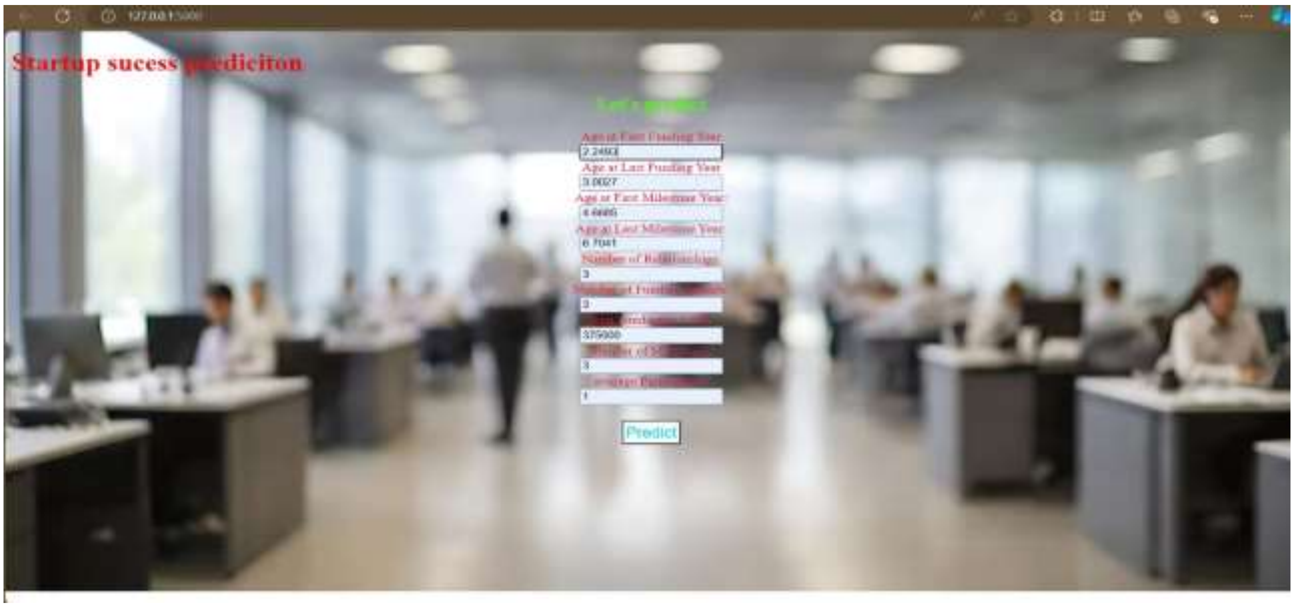
6.RESULT


Startup sucess prediciton

Let's predict

Age at First Funding Year
Enter first funding year
Age at Last Funding Year
Enter last funding year
Age at First Milestone Year
Enter first Milestone year
Age at Last Milestone Year
Enter last Milestone year
Number of Relationships
Enter relationships
Number of Funding [illegible]
Enter funding Rounds
[illegible]
Enter [illegible]
Number of [illegible]
Enter Milestones
[illegible]
Enter Participants

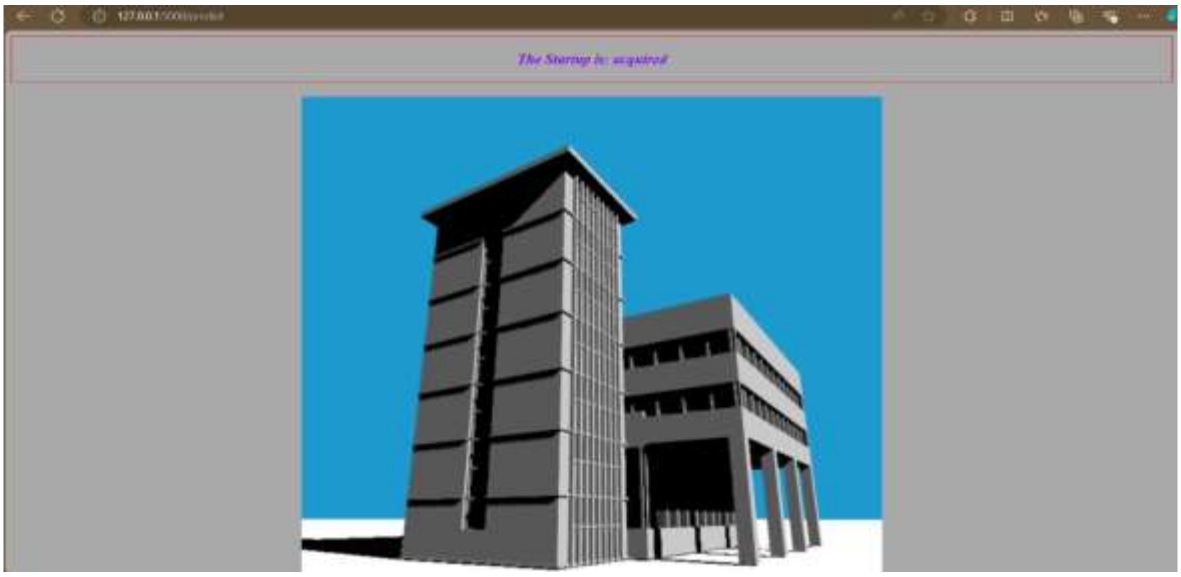
Predict


Startup sucess prediciton

Let's predict

Age at First Funding Year
2.2493
Age at Last Funding Year
3.0027
Age at First Milestone Year
4.6685
Age at Last Milestone Year
6.7041
Number of Relationships
3
Number of Funding [illegible]
3
[illegible]
375000
Number of [illegible]
3
[illegible]
1

Predict


The Startup is: acquired

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Data-Driven Insights:** Machine learning models analyze large datasets to uncover patterns and correlations, providing deep insights into the factors that drive startup success.

2. **Objective Evaluation**: ML models offer consistent and unbiased evaluations, reducing human bias in the assessment process.

3. **Predictive Accuracy:** By leveraging historical data and various metrics, advanced ML algorithms can accurately predict the success potential of startups.

4. **Early Risk Identification:** These models can identify potential risks and challenges early on, enabling startups to take proactive measures to mitigate issues.

5. **Resource Optimization:** By pinpointing the most promising startups, investors and stakeholders can allocate resources more effectively, ensuring optimal use of funding, mentorship, and support.

DISADVANTAGES:

1. **Data Quality and Availability:** Accurate predictions rely heavily on high-quality and comprehensive data. Many startups may not have sufficient historical data, or the available data might be incomplete, inaccurate, or biased, leading to unreliable predictions.

2. **Model Complexity and Interpretability:** Machine learning models, especially complex ones like deep learning, can be difficult to interpret. This lack of transparency can make it challenging for stakeholders to trust and understand the predictions, potentially limiting their usefulness in decision-making.

3. **Overfitting**: Machine learning models can sometimes overfit to the training data, meaning they perform well on historical data but poorly on unseen data. This is particularly problematic in the volatile and dynamic environment of startups, where past trends may not reliably predict future outcomes.

4. **Dynamic Market Conditions:** Startups operate in rapidly changing markets. Models trained on past data might not be able to adapt quickly to new market conditions, emerging trends, or disruptive technologies, reducing the accuracy and relevance of predictions.

5. **Bias and Fairness business implications. :** If the training data contains biases, the predictions made by the model can perpetuate or even amplify these biases. This can lead to unfair outcomes, such as favoring certain types of startups or founders over others, which can have significant ethical and business implications.

8.APPLICATIONS

1. Investor Decision-Making: Investors can use ML-driven success predictions to identify promising startups, helping them allocate funds more effectively and potentially increasing their return on investment.

2. Risk Management: Financial institutions and venture capitalists can assess the risk associated with funding a startup, allowing them to develop strategies to mitigate potential losses.

3. Resource Allocation:. Startups themselves can use predictions to make informed decisions about where to allocate resources, such as marketing budgets, hiring, and product development efforts, to maximize their chances of success

4. Mentorship and Support Programs: Incubators and accelerators can use predictions to identify startups that would benefit most from mentorship and resources, optimizing their support programs to nurture high-potential ventures.

5. Market Research and Trend Analysis: ML models can analyze market trends and predict which sectors or types of startups are likely to succeed, helping entrepreneurs and investors stay ahead of the curve and capitalize on emerging opportunities.

9.CONCLUSION

- The conclusion of "Prosperity Prognosticator: Startup Success Prediction Using Machine Learning" emphasizes the transformative impact of machine learning in the startup ecosystem. By integrating diverse data sources such as financial metrics, market conditions, team composition, and social media activity, the model captures the complex dynamics influencing startup success. Among various algorithms tested, ensemble methods like random forests proved to be the most effective, offering superior predictive accuracy and robustness.

- The research highlights the practical implications of these findings, suggesting that the model can significantly aid venture capitalists, incubators, and entrepreneurs by providing more reliable forecasts, thus enhancing decision-making processes. The scalability and adaptability of the model ensure its applicability across different regions and industries, allowing for continuous updates as more data becomes available, thereby maintaining its relevance and accuracy over time.

- In summary, the Prosperity Prognosticator showcases the potential of machine learning to revolutionize how startup success is predicted, fostering a more strategic and data-driven approach to investment and resource allocation. This paves the way for better identification of high-potential ventures, ultimately contributing to innovation and growth in the entrepreneurial landscape.

- Future enhancements could involve incorporating additional data sources and refining algorithms to further improve predictive precision.

10.FUTURE SCOPE

The future scope of prosperity prognosticator startups, which predict success using machine learning (ML), is promising due to several advancements and trends:

1.Enhanced Predictive Models:-

- Improved Algorithms: Development of more sophisticated algorithms (e.g., deep learning, ensemble methods) will enhance prediction accuracy.
- Feature Engineering: Better techniques for identifying and engineering relevant features from startup data (e.g., founder experience, market conditions) will improve model performance.

2. Big Data Utilization:

- Data Availability: Increasing amounts of data from various sources (e.g., social media, market trends, financial metrics) will allow for more comprehensive and accurate modeling.
- Real-time Analytics: Real-time data processing capabilities will enable continuous updates to predictive models, reflecting the latest market conditions and trends.

3. Integration with Business Processes:

- Decision Support: ML models can be integrated into decision-making tools, helping investors and entrepreneurs make informed decisions.
- Automation: Automation of certain predictive tasks can reduce human bias and increase the scalability of the predictive process.

4. Personalization:

- Tailored Predictions: Models can be personalized for specific industries or types of startups, improving the relevance and accuracy of predictions.
- Adaptive Learning: Continuous learning from new data will allow models to adapt to changing market dynamics and startup environments.

5. Regulatory and Ethical Considerations:

- Transparency and Accountability: Ensuring transparency in how predictions are made and accountability for the outcomes will be crucial for trust and adoption.
- Bias Mitigation: Efforts to identify and mitigate biases in predictive models will enhance fairness and equity.

6. Collaboration and Ecosystem Development:

- Partnerships: Collaborations between startups, academic institutions, and industry players can foster innovation and share valuable insights.
- Ecosystem Support: A supportive ecosystem that includes mentorship, funding, and infrastructure will enhance the practical application and success of ML-based predictive tools.

7. Economic Impact:

- Investment Optimization: Better predictions will lead to more efficient allocation of venture capital and resources, boosting overall startup success rates.
- Job Creation: As successful startups grow, they create jobs and contribute to economic development.

8. Global Reach:

- Cross-border Expansion: Predictive tools can help identify and nurture startups with potential for global expansion, contributing to a more interconnected and innovative global economy.
- Cultural Adaptation: Models can be adapted to account for cultural and regional differences, improving their applicability across diverse markets.

In summary, the future of startup success prediction using ML is bright, with potential for significant advancements in predictive precision, integration with business processes, and positive economic impact.

11.BIBILOGRAPHY

1.Authors: Alexandre N. Baier, Miguel N. G. Santos, and Pedro A. C. Marques

Source: Journal of Business Research, Volume 133, 2021, Pages 163-173

Summary: This paper explores various machine learning models to predict the success of startups using factors like team composition, financial performance, and market conditions.

"Success in Innovation: Evidence from European Start-Up Companies"

2.Authors: Cristiano Antonelli and Andrea Patrucco

Source: Research Policy, Volume 45, Issue 7, 2016, Pages 1308-1321

Summary: Analyzes the role of innovation, funding, and networking in the success of European startups.

"Factors Influencing Startup Success: A Case Study Analysis"

3.Authors: Rakesh Rathi, Rajesh Kumar, and Anand Kumar

Source: International Journal of Engineering Business Management, 2020

Summary: Investigates key factors like market demand, product uniqueness, and team expertise through case studies of successful startups.

"Machine Learning in Business: Applications of Predictive Analytics in Startups"

4.Authors: Karl R. Schmedders and Robert F. Freund

Source: Harvard Business Review, 2019

Summary: Discusses how predictive analytics can be used to assess startup potential, focusing on data-driven decision-making.

"The Role of Social Media in Startup Success: An Empirical Study"

5.Authors: Sarah J. Connell and Mark D. Weaver

Source: Journal of Small Business Management, Volume 57, Issue 3, 2019, Pages 1234-1256

Summary: Explores the impact of social media presence and engagement on the success rates of startups.

"Entrepreneurial Ecosystems and the Success of Startups: The Role of Network Effects"

6.Authors: Paul M. Vaaler and Scott D. Shane

Source: Management Science, Volume 64, Issue 1, 2018, Pages 284-300

Summary: Examines how the entrepreneurial ecosystem, including access to capital, mentorship, and networking opportunities, affects startup success.

"Evaluating Startup Success with Financial Ratios and Industry Benchmarks"

7.Authors: Emily D. Roberts and James A. Morgan

Source: Journal of Finance, Volume 74, Issue 2, 2020, Pages 567-589

Summary: Focuses on financial metrics and industry benchmarks as predictors of startup success.

"From Startup to Scale-Up: Factors Influencing Growth Trajectories"

8.Authors: Daniel J. Isenberg and Rebecca S. Fish

Source: MIT Sloan Management Review, 2020

Summary: Investigates the transition from startup to scale-up, identifying key growth factors and potential challenges.

These references cover a range of methodologies and perspectives, providing a comprehensive overview of the factors and models used in predicting startup success.

12.APPENDIX

Model building :

1)Dataset

2)Google colab and VS code Application Building

1. HTML file (Index file, Predict file)

1. CSS file

2. Models in pickle format

SOURCE CODE:

INDEX.HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width",initial-scale=1.0">
    <link rel="stylesheet" href="style.css" />
    <title>Startup suceess prediciton</title>
    <style>
      body {
        background: url("static/2.jpeg") center;
        height: 100%;
        background-position: center;
        background-size: cover;
        background-repeat: no-repeat;
        position: sticky;
      }
      h1 {
        color: rgb(236, 11, 11);
      }
      .btn {
        margin-top: 20px;
        padding: 3px;
        background-color: azure;
        font-size: larger;
        color: rgb(17, 208, 214);
        cursor: pointer;
      }
      form {
        color: crimson;
        align-content: center;
        text-align: center;
      }
    </style>
  </head>
  <body>
```

```
    <h1>Startup sucess prediciton</h1>
    <h2 style="color: rgb(76, 245, 14); text-align: center">Let's predict</h2>
    <div class="inputs">
      <form action="{{ url_for('predict')}}" method="post">
        <label>Age at First Funding Year</label><br />
        <input
          type="text"
          name="age_first_funding_year"
          placeholder="Enter first funding year"
        /><br />
        <label>Age at Last Funding Year</label><br />
        <input
          type="text"
          name="age_last_funding_year"
          placeholder="Enter last funding year "
        /><br />
        <label>Age at First Milestone Year:</label><br />
        <input
          type="text"
          name="age_first_milestone_year"
          placeholder="Enter first Milestone year"
        /><br />
        <label>Age at Last Milestone Year:</label><br />
        <input
          type="text"
          name="age_last_milestone_year"
          placeholder="Enter last Milestone year"
        /><br />
        <label>Number of Relationships:</label><br />
        <input
          type="text"
          name="relationships"
          placeholder="Enter relationships"
        /><br />
        <label>Number of Funding Rounds:</label><br />
        <input
          type="text"
          name="funding_rounds"
          placeholder="Enter funding Rounds"
        /><br />
        <label>Total Funding(in USD):</label><br />
        <input
          type="text"
          name="funding_total_usd"
          placeholder="Enter rupees"
        /><br />
        <label>Number of Milestones:</label><br />
        <input
```

```
          type="text"
          name="milestones"
          placeholder="Enter Milestones"
        /><br />
        <label>average Participants:</label><br />
        <input
          type="text"
          name="avg_participants"
          placeholder="Enter Participants"
        /><br />
        <a href="result.html"
          ><button class="btn" type="submit">Predict</button></a
        >
      </form>
    </div>

    <br /><br />
    <section>
      <h3 style="color: blueviolet; text-align: center">
        {{ prediction_text }}
      </h3>
    </section>
  </body>
</html>
```

RESULT.HTML:

```
<html>
  <head>
    <title>result</title>
    <style>
      body {
        background-color: darkgrey;
      }
      .output {
        padding: 20px;
        border: 1px solid red;
        text-align: center;
        color: rgb(124, 0, 241);
        font-style: italic;
        font-size: larger;
      }
      .result {
        display: block;
        margin-left: auto;
        margin-right: auto;
        width: 50%;
      }
    </style>
  </head>
  <body>
    <h3 class="output">{{ prediction_text }}</h3>
    <img class="result" src="static/3.gif" alt="prediction" width="200" />
  </body>
</html>
```

STYLE.CSS

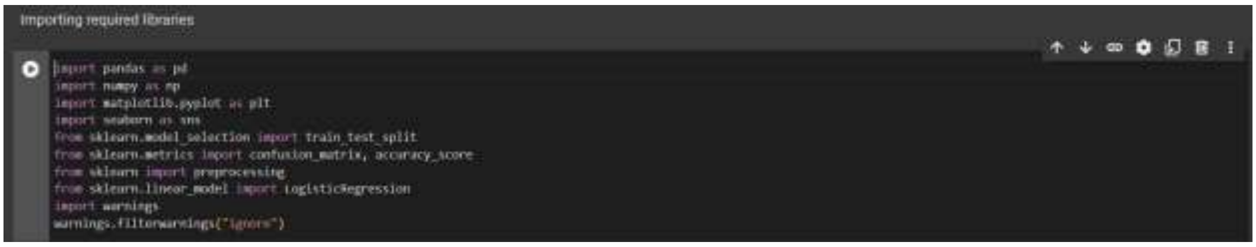
```css
h1 {
  text-decoration-color: blueviolet;
  color: chocolate;
}
div {
  align-content: center;
}
form {
  align-items: center;
  text-align: center;
}
h3 {
  text-align: center;
  color: #000000;
  border-spacing: center;
  background-color: aqua;
  text-decoration-color: blue;
  transition: ease-out;
}
```

APP.PY

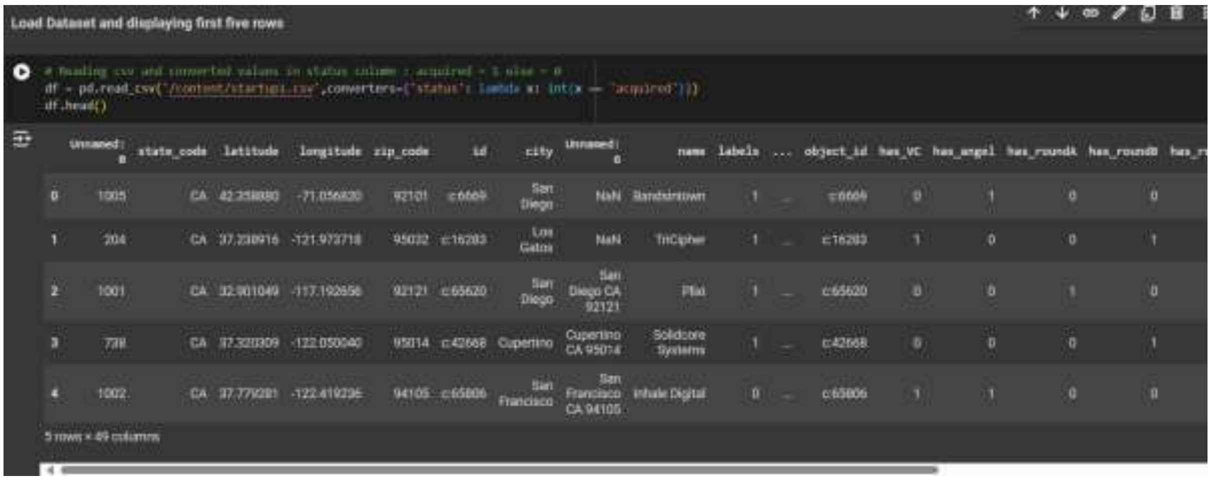
```python
from flask import Flask,render_template,request
import joblib
import numpy as np
import pandas as pd
#import pickle
app=Flask(__name__)
model=joblib.load('random_forest_model.pkl')
#model=pickle.load(open('random_forest_mode.pkl','rb'))
app=Flask(__name__,template_folder='template')
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
   input_feature=[x for x in request.form.values()]
   input_feature=np.transpose(input_feature)
   input_feature=[np.array(input_feature)]
   print(input_feature)
   names=['age_first_funding_year','age_last_funding_year','age_first_milestone_year','age_last_milestone_year','relationships','funding_rounds','funding_total_usd','milestones','avg_participants']
   data=pd.DataFrame(input_feature,columns=names)
   prediction=model.predict(data)
   result=int(prediction[0])
   print(result)
   if result==1:
      result='acquired'
   else:
      result='closed'
   return render_template('result.html', prediction_text='The Startup is: {}'.format(result))
if __name__=='__main__':
  app.run(debug=True)
```

CODE SNIPPETS:

MODEL BUILDING:


Importing required libraries

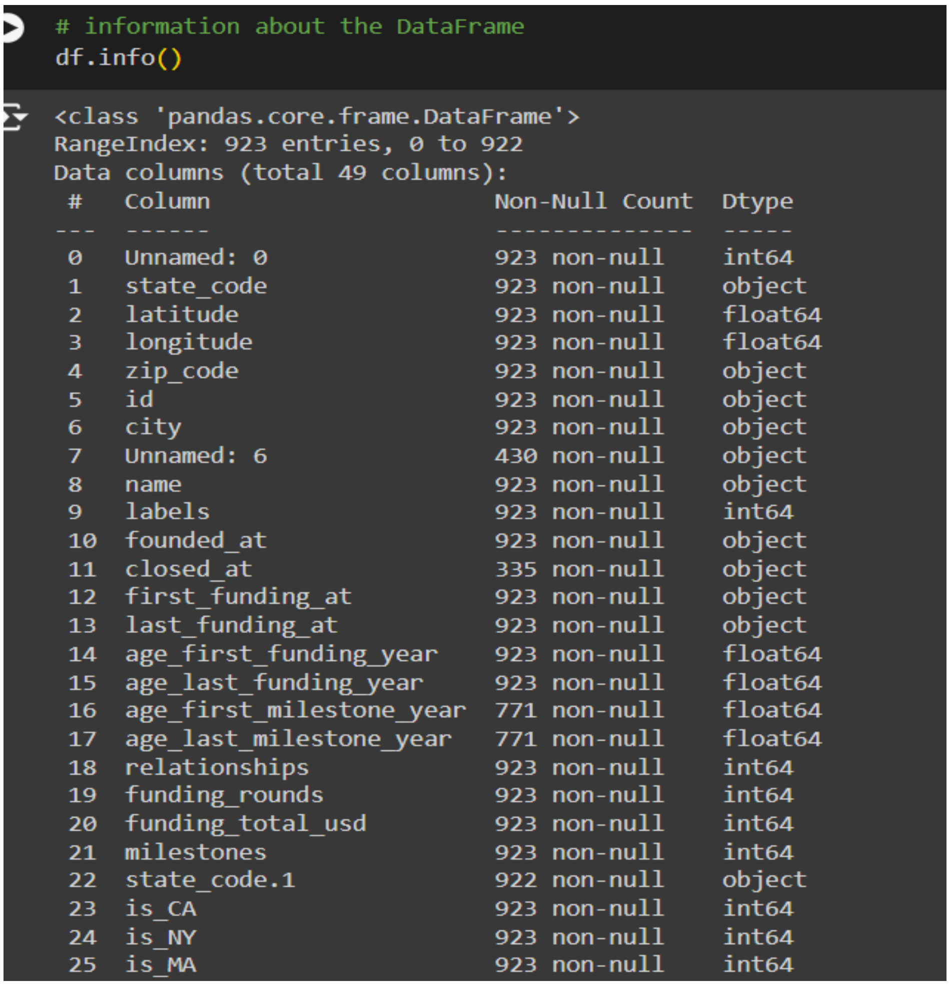
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings("ignore")
```


Load Dataset and displaying first five rows

```
# Reading csv and converted values in status column : acquired = 1 else = 0
df = pd.read_csv('/content/startups.csv',converters={'status': lambda x: int(x == 'acquired')})
df.head()
```

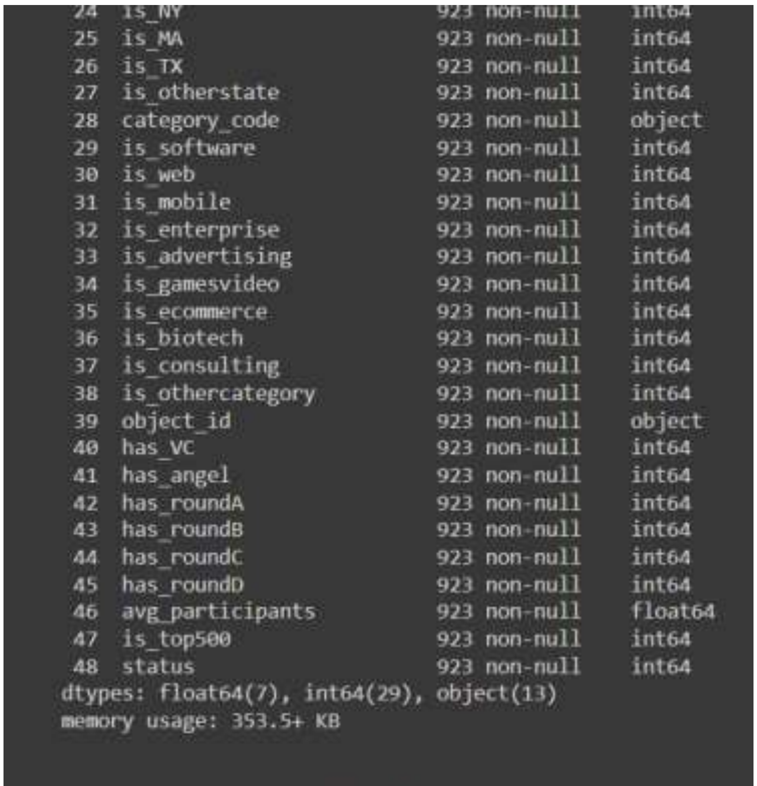
	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed: 6	name	labels	...	object_id	has_VC	has_angel	has_roundA	has_roundB	has_r
0	1005	CA	42.358880	-71.056820	92101	c:6669	San Diego	NaN	Bandsintown	1	...	c:6669	0	1	0	0	
1	204	CA	37.238916	-121.973718	95032	c:16283	Los Gatos	NaN	TriCipher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

5 rows × 49 columns



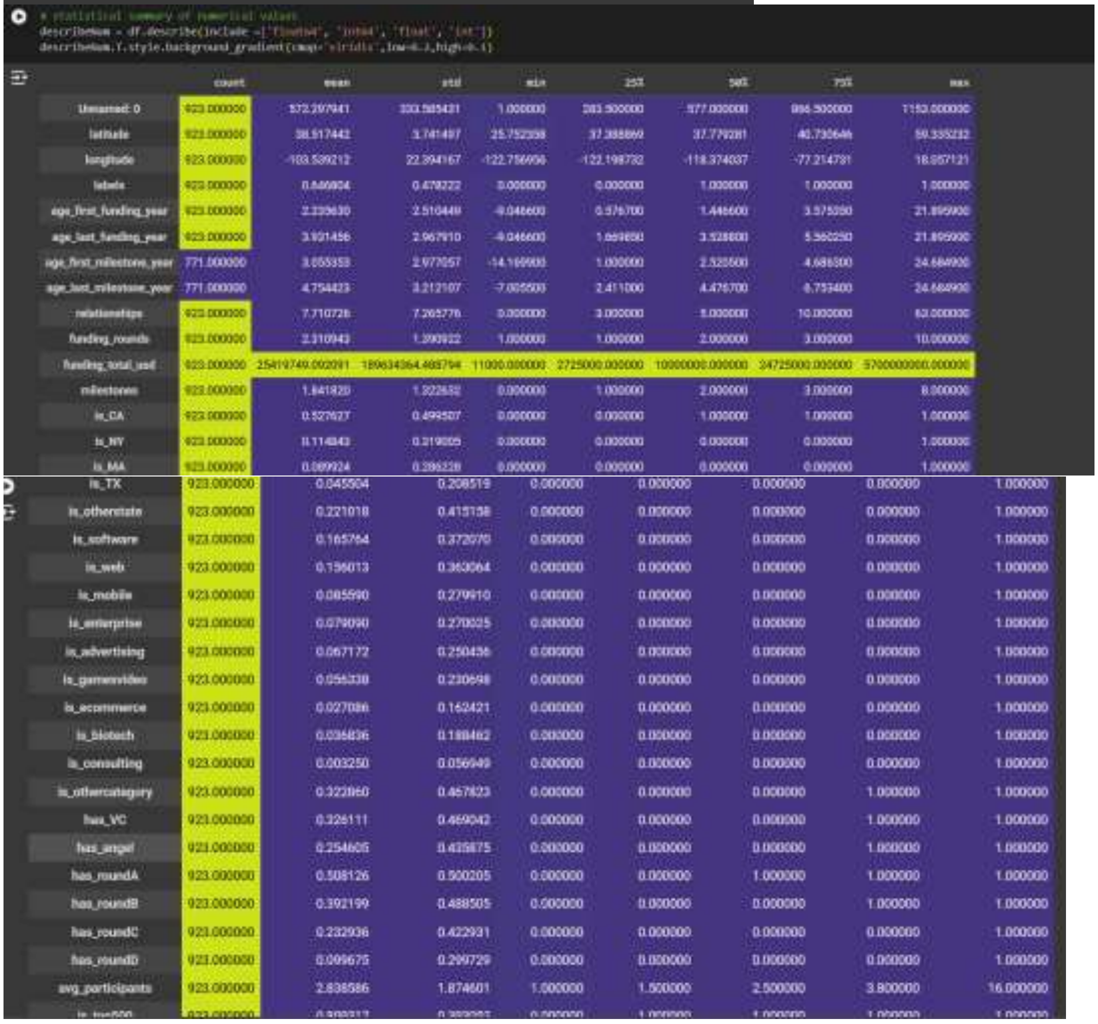
```
# information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 923 entries, 0 to 922
Data columns (total 49 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                923 non-null    int64
 1   state_code                923 non-null    object
 2   latitude                  923 non-null    float64
 3   longitude                 923 non-null    float64
 4   zip_code                  923 non-null    object
 5   id                        923 non-null    object
 6   city                      923 non-null    object
 7   Unnamed: 6                430 non-null    object
 8   name                      923 non-null    object
 9   labels                    923 non-null    int64
 10  founded_at                923 non-null    object
 11  closed_at                 335 non-null    object
 12  first_funding_at          923 non-null    object
 13  last_funding_at           923 non-null    object
 14  age_first_funding_year    923 non-null    float64
 15  age_last_funding_year     923 non-null    float64
 16  age_first_milestone_year  771 non-null    float64
 17  age_last_milestone_year   771 non-null    float64
 18  relationships             923 non-null    int64
 19  funding_rounds            923 non-null    int64
 20  funding_total_usd         923 non-null    int64
 21  milestones                923 non-null    int64
 22  state_code.1              922 non-null    object
 23  is_CA                     923 non-null    int64
 24  is_NY                     923 non-null    int64
 25  is_MA                     923 non-null    int64
```



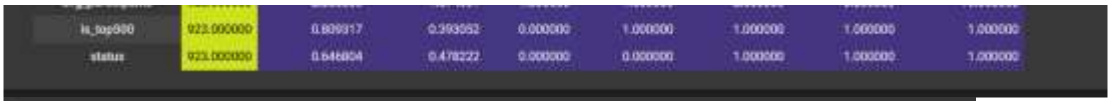
```
 24  is_NY                     923 non-null    int64
 25  is_MA                     923 non-null    int64
 26  is_TX                     923 non-null    int64
 27  is_otherstate             923 non-null    int64
 28  category_code             923 non-null    object
 29  is_software               923 non-null    int64
 30  is_web                    923 non-null    int64
 31  is_mobile                 923 non-null    int64
 32  is_enterprise             923 non-null    int64
 33  is_advertising            923 non-null    int64
 34  is_gamesvideo             923 non-null    int64
 35  is_ecommerce              923 non-null    int64
 36  is_biotech                923 non-null    int64
 37  is_consulting             923 non-null    int64
 38  is_othercategory          923 non-null    int64
 39  object_id                 923 non-null    object
 40  has_VC                    923 non-null    int64
 41  has_angel                 923 non-null    int64
 42  has_roundA                923 non-null    int64
 43  has_roundB                923 non-null    int64
 44  has_roundC                923 non-null    int64
 45  has_roundD                923 non-null    int64
 46  avg_participants          923 non-null    float64
 47  is_top500                 923 non-null    int64
 48  status                    923 non-null    int64
dtypes: float64(7), int64(29), object(13)
memory usage: 353.5+ KB
```

Statistical summary description of dataset

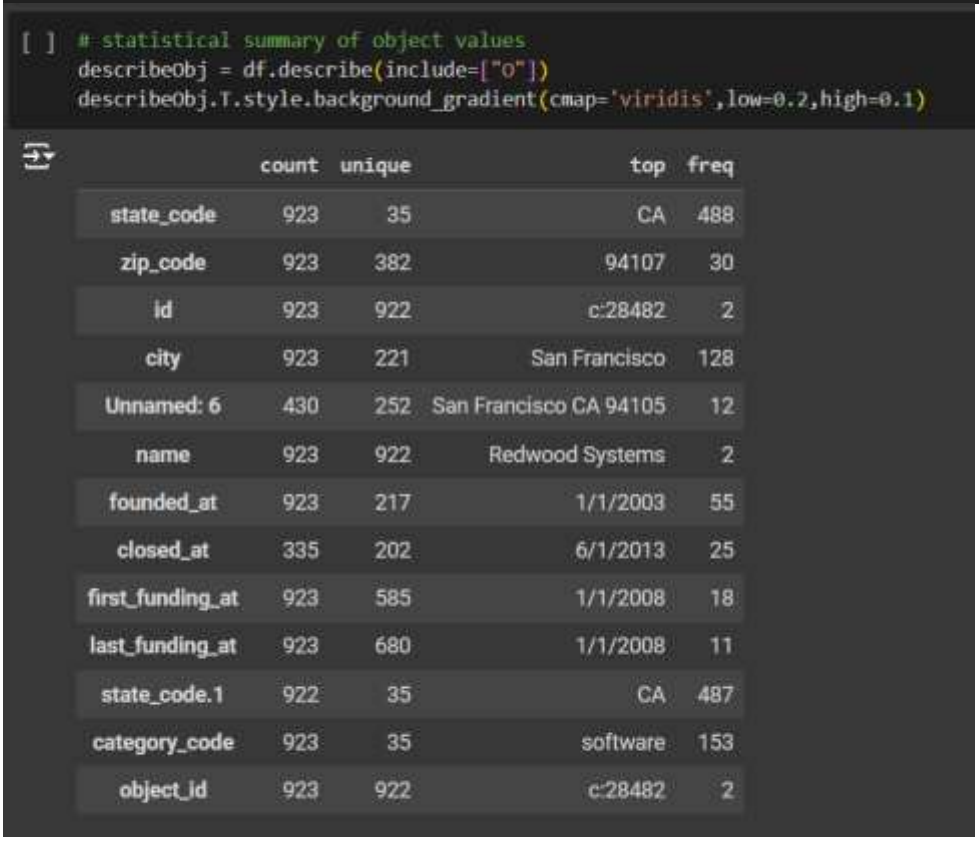


```
# statistical summary of numerical values
describeNum = df.describe(include =['float64', 'int64', 'float', 'int'])
describeNum.T.style.background_gradient(cmap='viridis',low=0.2,high=0.1)
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	923.000000	572.297941	333.585431	1.000000	283.500000	577.000000	866.500000	1153.000000
latitude	923.000000	38.517442	3.741497	25.752358	37.388869	37.779281	40.730646	59.335232
longitude	923.000000	-103.539212	22.394167	-122.756956	-122.198732	-118.374037	-77.214731	18.057121
labels	923.000000	0.646804	0.478222	0.000000	0.000000	1.000000	1.000000	1.000000
age_first_funding_year	923.000000	2.235630	2.510449	-9.046600	0.576700	1.446600	3.575350	21.895900
age_last_funding_year	923.000000	3.931456	2.967910	-9.046600	1.669850	3.528800	5.560250	21.895900
age_first_milestone_year	771.000000	3.055353	2.977057	-14.169900	1.000000	2.520500	4.686300	24.684900
age_last_milestone_year	771.000000	4.754423	3.212107	-7.005500	2.411000	4.476700	6.753400	24.684900
relationships	923.000000	7.710726	7.265776	0.000000	3.000000	5.000000	10.000000	63.000000
funding_rounds	923.000000	2.310943	1.390922	1.000000	1.000000	2.000000	3.000000	10.000000
funding_total_usd	923.000000	25419749.092091	189634364.489794	11000.000000	2725000.000000	10000000.000000	24725000.000000	5700000000.000000
milestones	923.000000	1.841820	1.322632	0.000000	1.000000	2.000000	3.000000	8.000000
is_CA	923.000000	0.527627	0.499507	0.000000	0.000000	1.000000	1.000000	1.000000
is_NY	923.000000	0.114843	0.319005	0.000000	0.000000	0.000000	0.000000	1.000000
is_MA	923.000000	0.089924	0.286228	0.000000	0.000000	0.000000	0.000000	1.000000
is_TX	923.000000	0.045504	0.208519	0.000000	0.000000	0.000000	0.000000	1.000000
is_otherstate	923.000000	0.221018	0.415158	0.000000	0.000000	0.000000	0.000000	1.000000
is_software	923.000000	0.165764	0.372070	0.000000	0.000000	0.000000	0.000000	1.000000
is_web	923.000000	0.156013	0.363064	0.000000	0.000000	0.000000	0.000000	1.000000
is_mobile	923.000000	0.085590	0.279910	0.000000	0.000000	0.000000	0.000000	1.000000
is_enterprise	923.000000	0.079090	0.270025	0.000000	0.000000	0.000000	0.000000	1.000000
is_advertising	923.000000	0.067172	0.250436	0.000000	0.000000	0.000000	0.000000	1.000000
is_gamesvideo	923.000000	0.056338	0.230698	0.000000	0.000000	0.000000	0.000000	1.000000
is_ecommerce	923.000000	0.027086	0.162421	0.000000	0.000000	0.000000	0.000000	1.000000
is_biotech	923.000000	0.036836	0.188462	0.000000	0.000000	0.000000	0.000000	1.000000
is_consulting	923.000000	0.003250	0.056949	0.000000	0.000000	0.000000	0.000000	1.000000
is_othercategory	923.000000	0.322860	0.467823	0.000000	0.000000	0.000000	1.000000	1.000000
has_VC	923.000000	0.326111	0.469042	0.000000	0.000000	0.000000	1.000000	1.000000
has_angel	923.000000	0.254605	0.435875	0.000000	0.000000	0.000000	1.000000	1.000000
has_roundA	923.000000	0.508126	0.500205	0.000000	0.000000	1.000000	1.000000	1.000000
has_roundB	923.000000	0.392199	0.488505	0.000000	0.000000	0.000000	1.000000	1.000000
has_roundC	923.000000	0.232936	0.422931	0.000000	0.000000	0.000000	0.000000	1.000000
has_roundD	923.000000	0.099675	0.299729	0.000000	0.000000	0.000000	0.000000	1.000000
avg_participants	923.000000	2.838586	1.874601	1.000000	1.500000	2.500000	3.800000	16.000000
is_top500	923.000000	[illegible]	[illegible]	0.000000	1.000000	1.000000	1.000000	1.000000

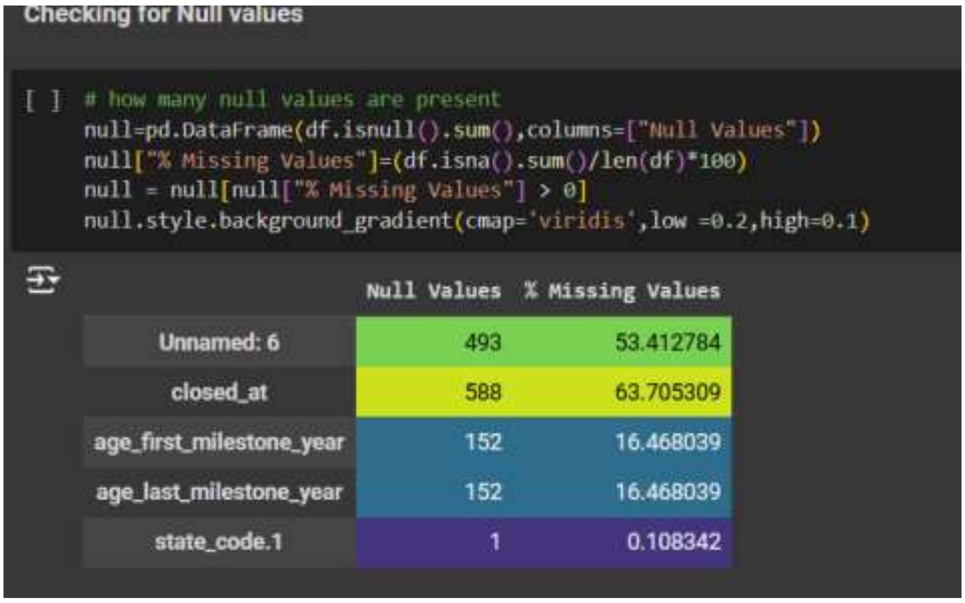


is_top500	923.000000	0.809317	0.393052	0.000000	1.000000	1.000000	1.000000	1.000000
status	923.000000	0.646804	0.478222	0.000000	0.000000	1.000000	1.000000	1.000000



```
[ ] # statistical summary of object values
    describeObj = df.describe(include=["O"])
    describeObj.T.style.background_gradient(cmap='viridis',low=0.2,high=0.1)
```

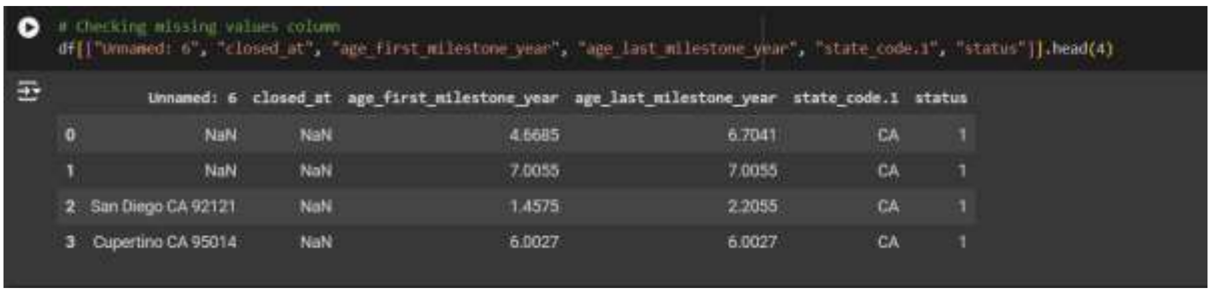
	count	unique	top	freq
state_code	923	35	CA	488
zip_code	923	382	94107	30
id	923	922	c:28482	2
city	923	221	San Francisco	128
Unnamed: 6	430	252	San Francisco CA 94105	12
name	923	922	Redwood Systems	2
founded_at	923	217	1/1/2003	55
closed_at	335	202	6/1/2013	25
first_funding_at	923	585	1/1/2008	18
last_funding_at	923	680	1/1/2008	11
state_code.1	922	35	CA	487
category_code	923	35	software	153
object_id	923	922	c:28482	2



Checking for Null values

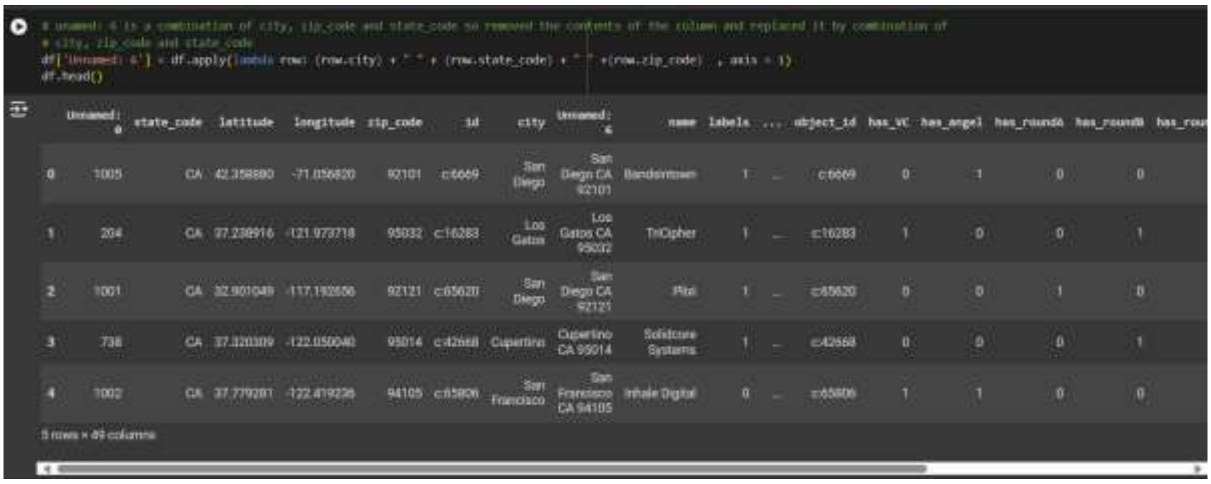
```
[ ] # how many null values are present
    null=pd.DataFrame(df.isnull().sum(),columns=["Null Values"])
    null["% Missing Values"]=(df.isna().sum()/len(df)*100)
    null = null[null["% Missing Values"] > 0]
    null.style.background_gradient(cmap='viridis',low =0.2,high=0.1)
```

	Null Values	% Missing Values
Unnamed: 6	493	53.412784
closed_at	588	63.705309
age_first_milestone_year	152	16.468039
age_last_milestone_year	152	16.468039
state_code.1	1	0.108342



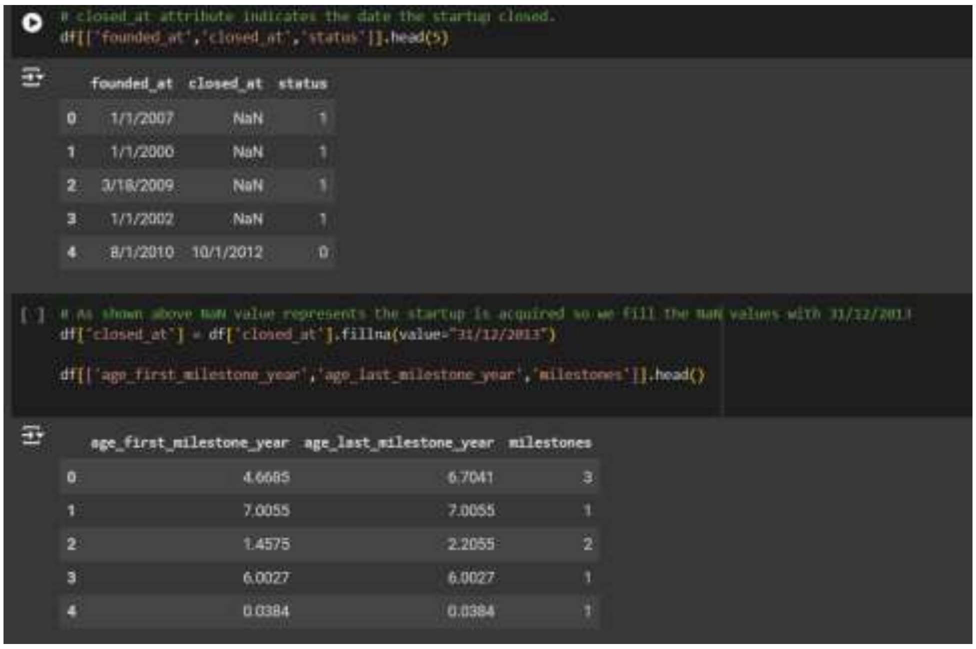
```
# Checking missing values column
df[["Unnamed: 6", "closed_at", "age_first_milestone_year", "age_last_milestone_year", "state_code.1", "status"]].head(4)
```

	Unnamed: 6	closed_at	age_first_milestone_year	age_last_milestone_year	state_code.1	status
0	NaN	NaN	4.6685	6.7041	CA	1
1	NaN	NaN	7.0055	7.0055	CA	1
2	San Diego CA 92121	NaN	1.4575	2.2055	CA	1
3	Cupertino CA 95014	NaN	6.0027	6.0027	CA	1



```python
# unamed: 6 is a combination of city, zip_code and state_code so removed the contents of the column and replaced it by combination of
# city, zip_code and state_code
df['Unnamed: 6'] = df.apply(lambda row: (row.city) + " " + (row.state_code) + " " +(row.zip_code)  , axis = 1)
df.head()
```

```
   Unnamed:  state_code  latitude   longitude  zip_code  id       city       Unnamed:      name           labels ... object_id  has_VC  has_angel  has_roundA  has_roundB  has_rou
       0                                                                         6
0  1005      CA          42.358880  -71.056820  92101    c:6669   San        San           Bandsintown    1      ...  c:6669    0       1          0           0
                                                                  Diego      Diego CA
                                                                             92101
1  204       CA          37.238916  -121.973718 95032    c:16283  Los        Los           TriCipher      1      ...  c:16283   1       0          0           1
                                                                  Gatos      Gatos CA
                                                                             95032
2  1001      CA          32.901049  -117.192656 92121    c:65620  San        San           Plixi          1      ...  c:65620   0       0          1           0
                                                                  Diego      Diego CA
                                                                             92121
3  738       CA          37.320309  -122.050040 95014    c:42668  Cupertino  Cupertino     Solidcore      1      ...  c:42668   0       0          0           1
                                                                             CA 95014      Systems
4  1002      CA          37.779281  -122.419236 94105    c:65806  San        San           Inhale Digital 0      ...  c:65806   1       1          0           0
                                                                  Francisco  Francisco
                                                                             CA 94105
5 rows × 49 columns
```

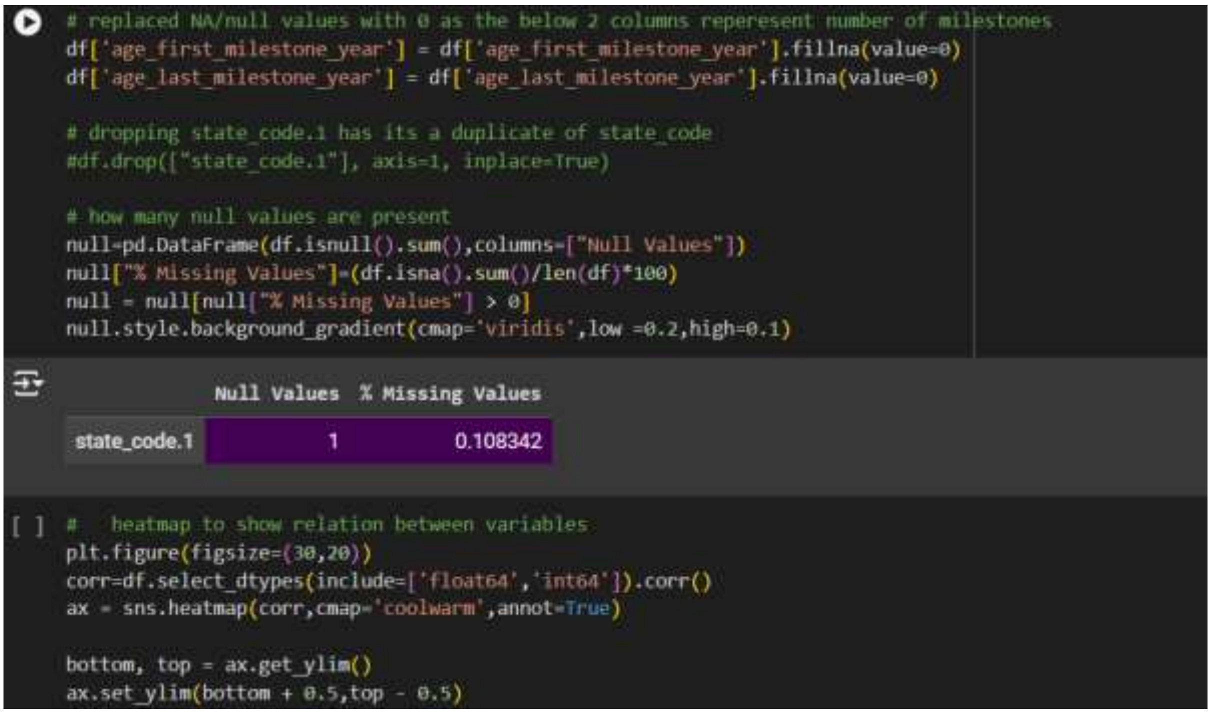


```python
# closed_at attribute indicates the date the startup closed.
df[['founded_at','closed_at','status']].head(5)
```

```
   founded_at  closed_at  status
0  1/1/2007    NaN        1
1  1/1/2000    NaN        1
2  3/18/2009   NaN        1
3  1/1/2002    NaN        1
4  8/1/2010    10/1/2012  0
```

```python
[ ] # As shown above NaN value represents the startup is acquired so we fill the NaN values with 31/12/2013
    df['closed_at'] = df['closed_at'].fillna(value="31/12/2013")

    df[['age_first_milestone_year','age_last_milestone_year','milestones']].head()
```

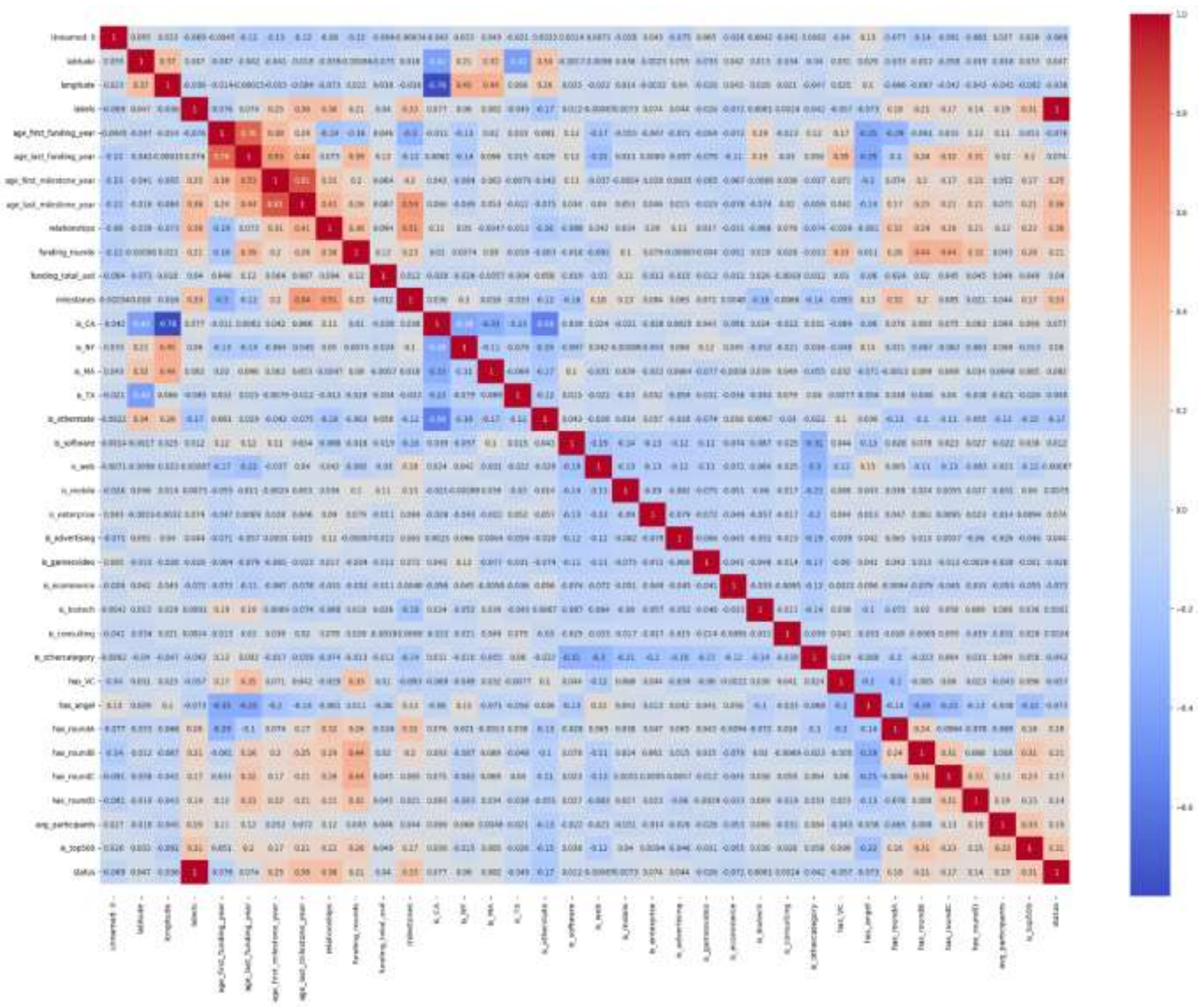
```
   age_first_milestone_year  age_last_milestone_year  milestones
0                    4.6685                   6.7041           3
1                    7.0055                   7.0055           1
2                    1.4575                   2.2055           2
3                    6.0027                   6.0027           1
4                    0.0384                   0.0384           1
```

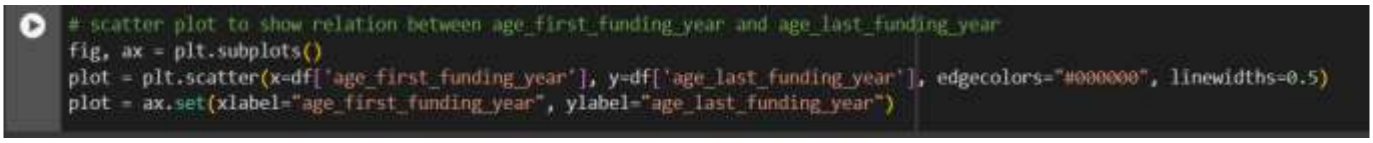


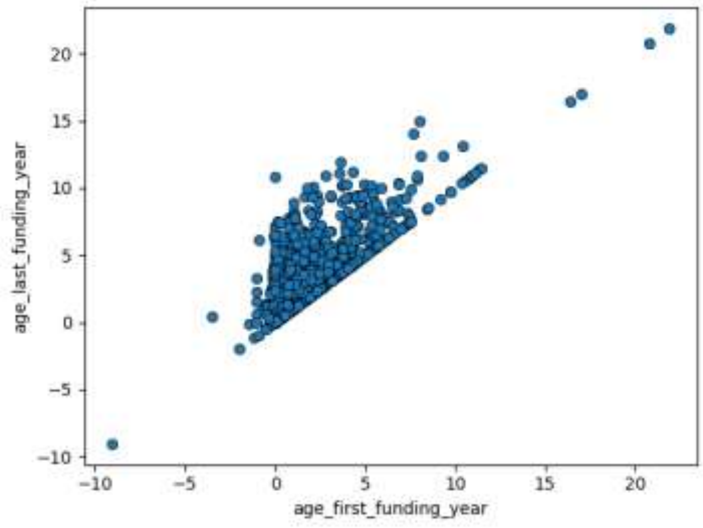
```python
# replaced NA/null values with 0 as the below 2 columns reperesent number of milestones
df['age_first_milestone_year'] = df['age_first_milestone_year'].fillna(value=0)
df['age_last_milestone_year'] = df['age_last_milestone_year'].fillna(value=0)

# dropping state_code.1 has its a duplicate of state_code
#df.drop(["state_code.1"], axis=1, inplace=True)

# how many null values are present
null=pd.DataFrame(df.isnull().sum(),columns=["Null Values"])
null["% Missing Values"]=(df.isna().sum()/len(df)*100)
null = null[null["% Missing Values"] > 0]
null.style.background_gradient(cmap='viridis',low =0.2,high=0.1)
```

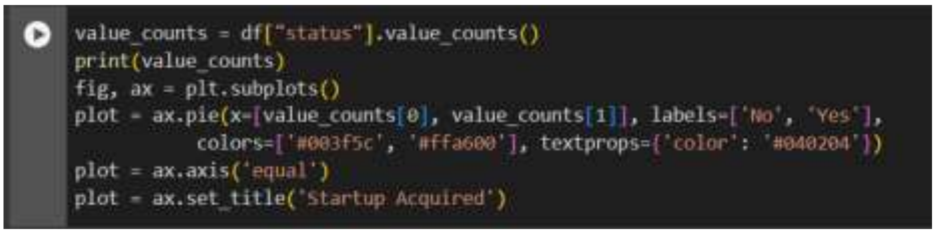
```
              Null Values  % Missing Values
state_code.1            1          0.108342
```

```python
[ ] #   heatmap to show relation between variables
    plt.figure(figsize=(30,20))
    corr=df.select_dtypes(include=['float64','int64']).corr()
    ax = sns.heatmap(corr,cmap='coolwarm',annot=True)

    bottom, top = ax.get_ylim()
    ax.set_ylim(bottom + 0.5,top - 0.5)
```

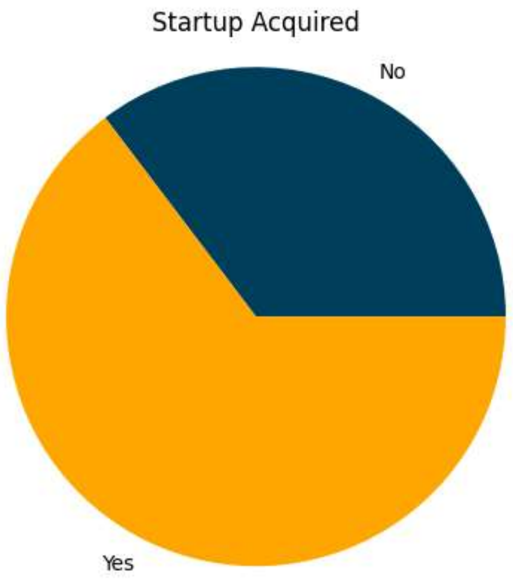


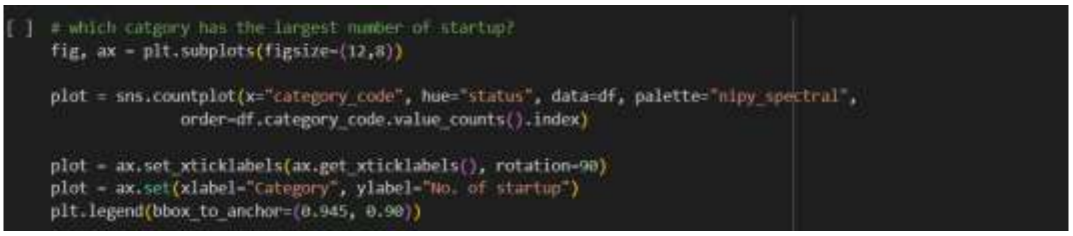

```
# scatter plot to show relation between age_first_funding_year and age_last_funding_year
fig, ax = plt.subplots()
plot = plt.scatter(x=df['age_first_funding_year'], y=df['age_last_funding_year'], edgecolors="#000000", linewidths=0.5)
plot = ax.set(xlabel="age_first_funding_year", ylabel="age_last_funding_year")
```



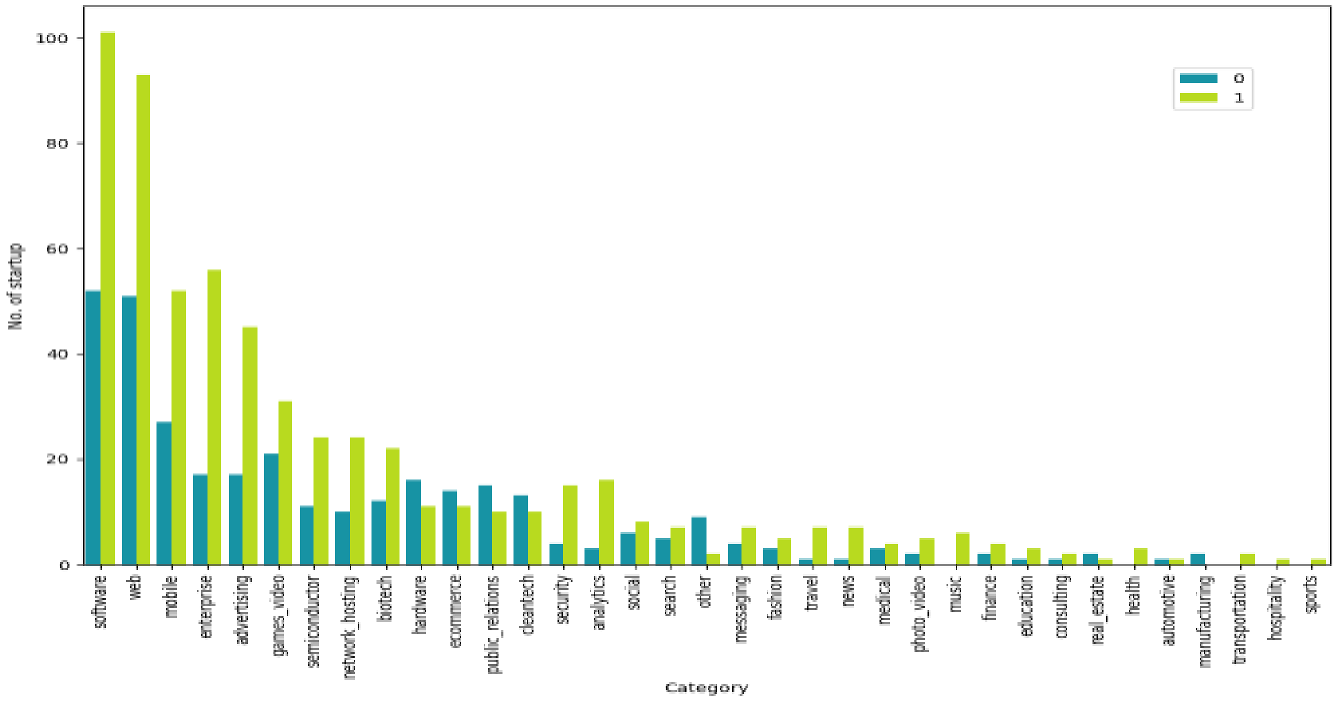


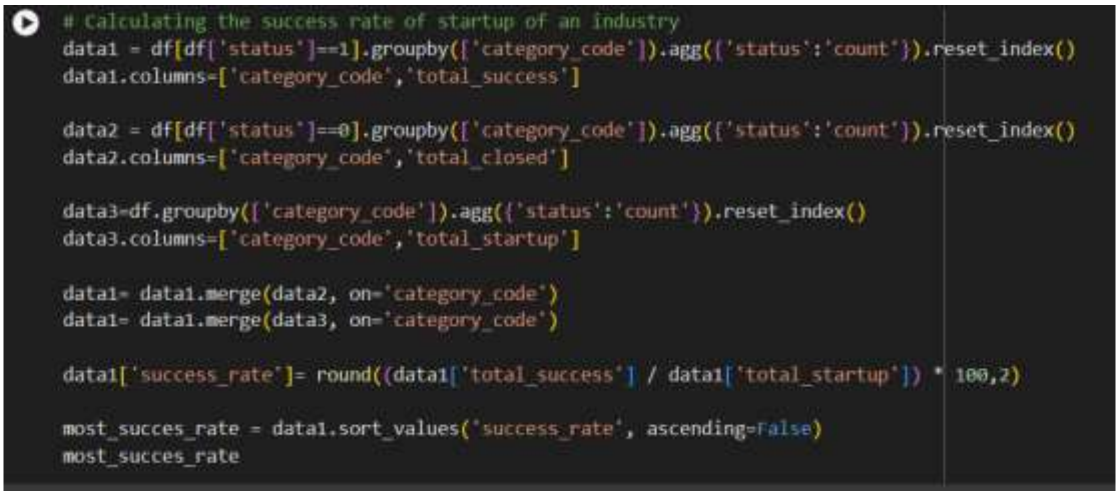
```
value_counts = df["status"].value_counts()
print(value_counts)
fig, ax = plt.subplots()
plot = ax.pie(x=[value_counts[0], value_counts[1]], labels=['No', 'Yes'],
              colors=['#003f5c', '#ffa600'], textprops={'color': '#040204'})
plot = ax.axis('equal')
plot = ax.set_title('Startup Acquired')
```



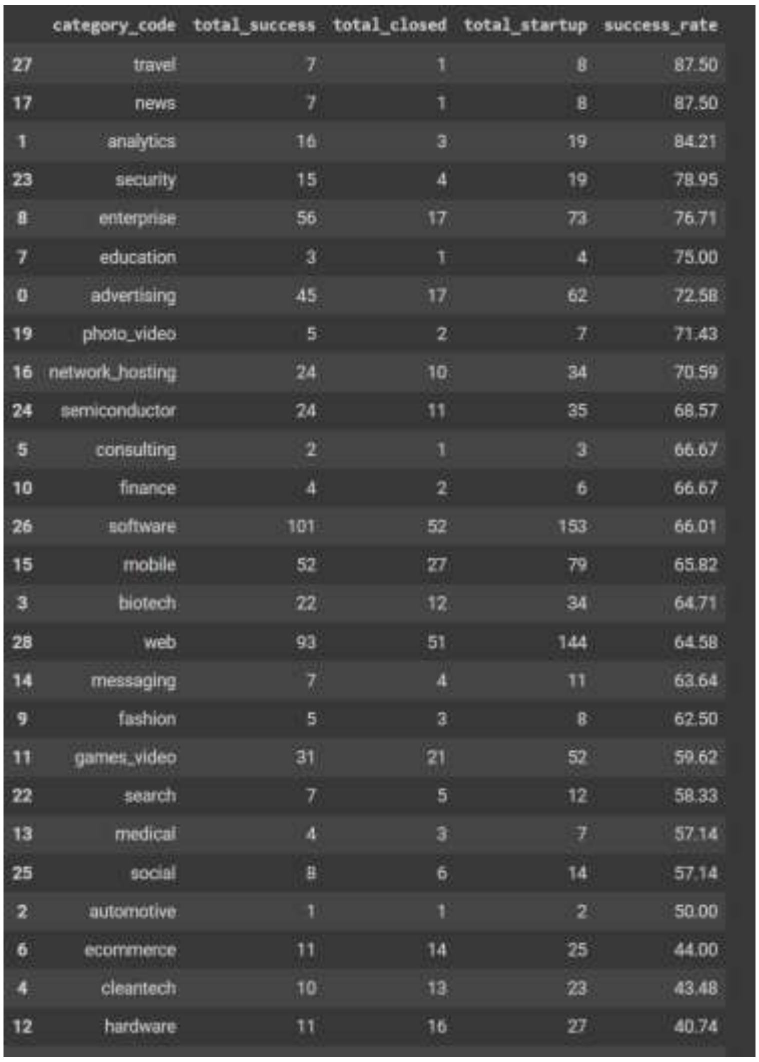


```
[ ] # which catgory has the largest number of startup?
    fig, ax = plt.subplots(figsize=(12,8))

    plot = sns.countplot(x="category_code", hue="status", data=df, palette="nipy_spectral",
                order=df.category_code.value_counts().index)

    plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    plot = ax.set(xlabel="Category", ylabel="No. of startup")
    plt.legend(bbox_to_anchor=(0.945, 0.90))
```

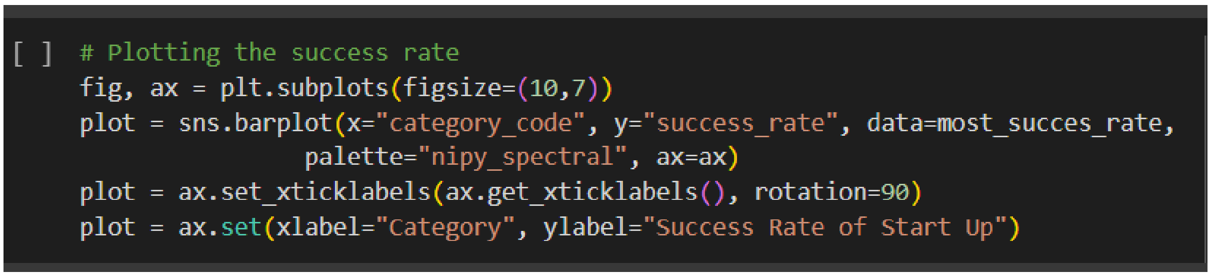




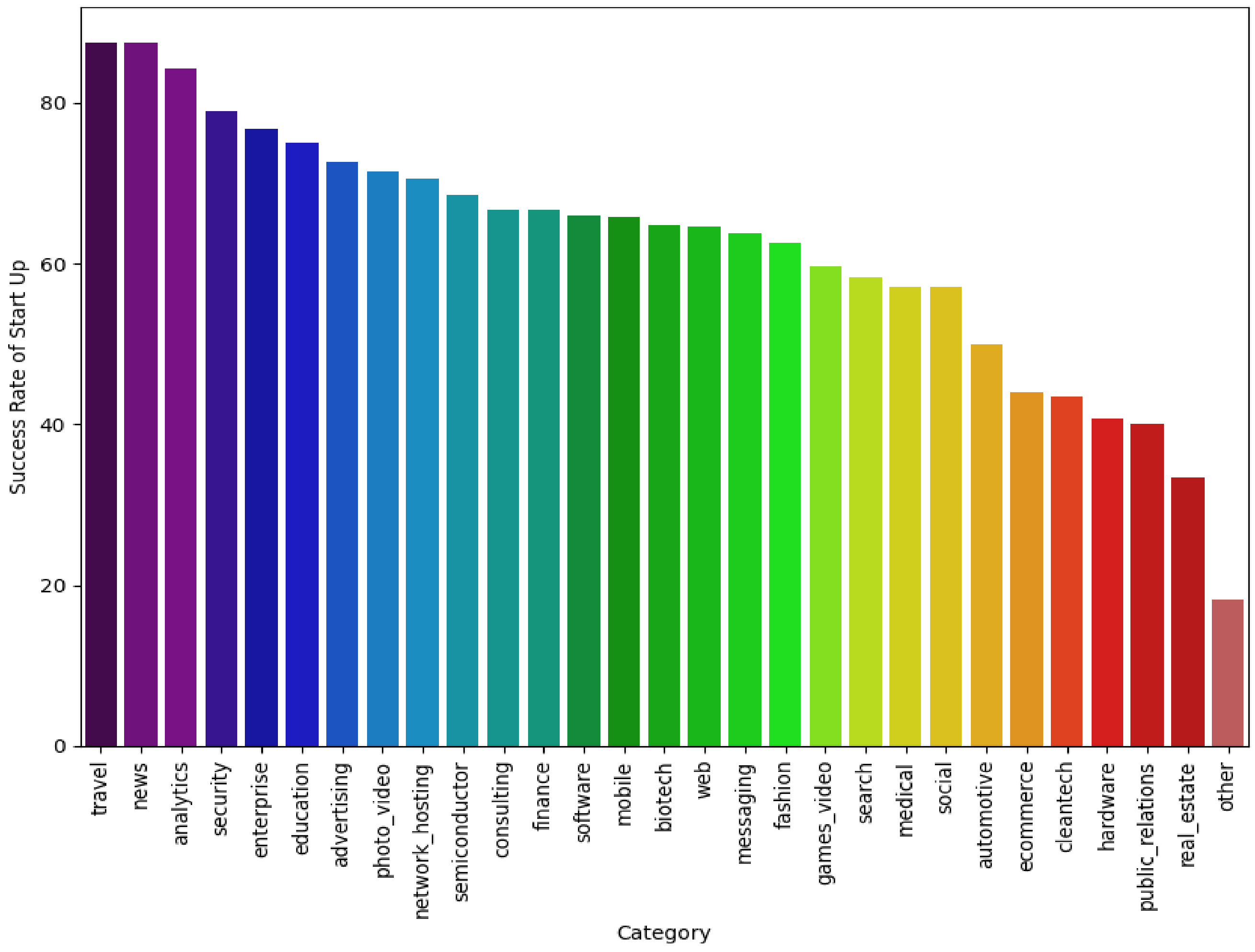
```python
# Calculating the success rate of startup of an industry
data1 = df[df['status']==1].groupby(['category_code']).agg({'status':'count'}).reset_index()
data1.columns=['category_code','total_success']

data2 = df[df['status']==0].groupby(['category_code']).agg({'status':'count'}).reset_index()
data2.columns=['category_code','total_closed']

data3=df.groupby(['category_code']).agg({'status':'count'}).reset_index()
data3.columns=['category_code','total_startup']

data1= data1.merge(data2, on='category_code')
data1= data1.merge(data3, on='category_code')

data1['success_rate']= round((data1['total_success'] / data1['total_startup']) * 100,2)

most_succes_rate = data1.sort_values('success_rate', ascending=False)
most_succes_rate
```

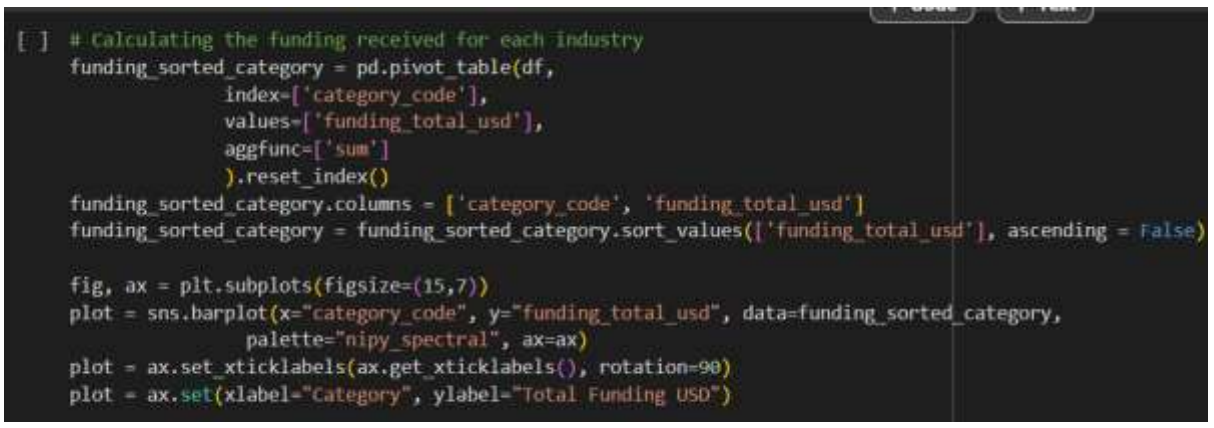


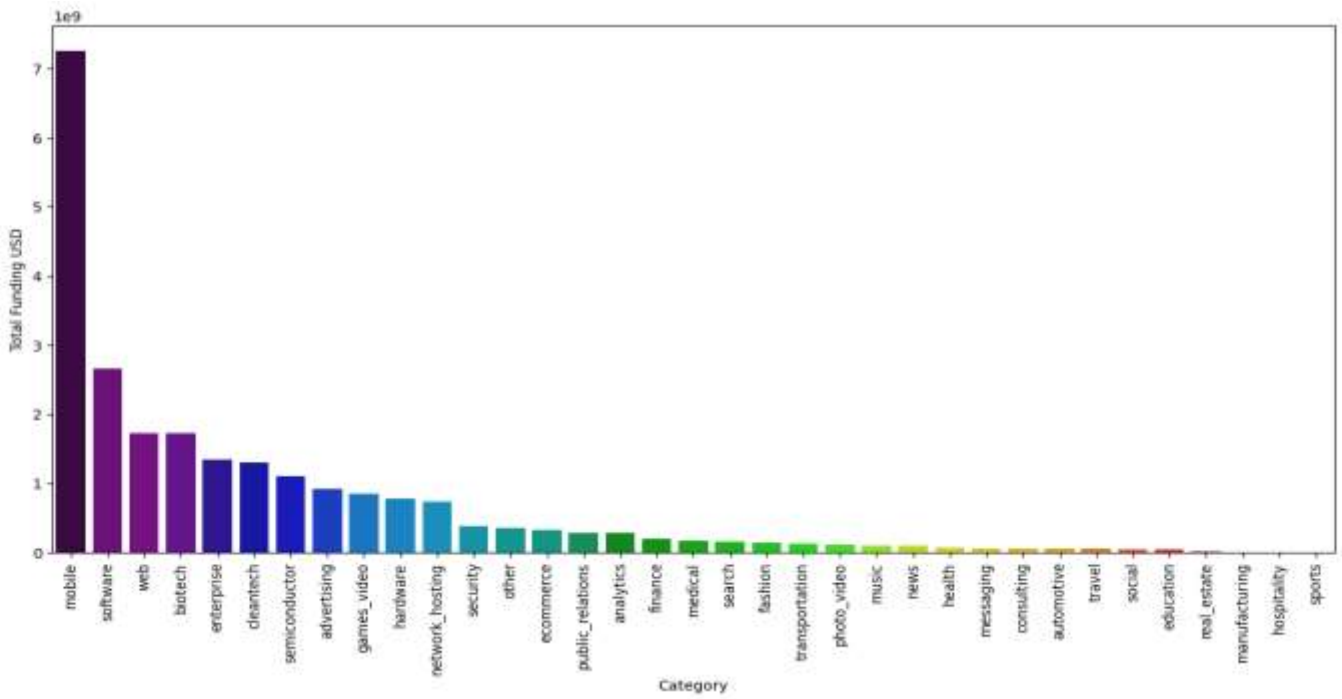
	category_code	total_success	total_closed	total_startup	success_rate
27	travel	7	1	8	87.50
17	news	7	1	8	87.50
1	analytics	16	3	19	84.21
23	security	15	4	19	78.95
8	enterprise	56	17	73	76.71
7	education	3	1	4	75.00
0	advertising	45	17	62	72.58
19	photo_video	5	2	7	71.43
16	network_hosting	24	10	34	70.59
24	semiconductor	24	11	35	68.57
5	consulting	2	1	3	66.67
10	finance	4	2	6	66.67
26	software	101	52	153	66.01
15	mobile	52	27	79	65.82
3	biotech	22	12	34	64.71
28	web	93	51	144	64.58
14	messaging	7	4	11	63.64
9	fashion	5	3	8	62.50
11	games_video	31	21	52	59.62
22	search	7	5	12	58.33
13	medical	4	3	7	57.14
25	social	8	6	14	57.14
2	automotive	1	1	2	50.00
6	ecommerce	11	14	25	44.00
4	cleantech	10	13	23	43.48
12	hardware	11	16	27	40.74

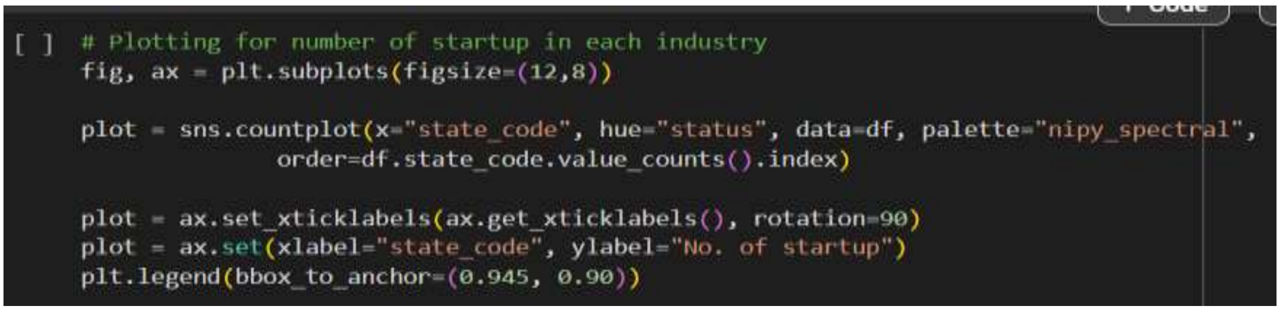


```python
[ ]  # Plotting the success rate
     fig, ax = plt.subplots(figsize=(10,7))
     plot = sns.barplot(x="category_code", y="success_rate", data=most_succes_rate,
                        palette="nipy_spectral", ax=ax)
     plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
     plot = ax.set(xlabel="Category", ylabel="Success Rate of Start Up")
```

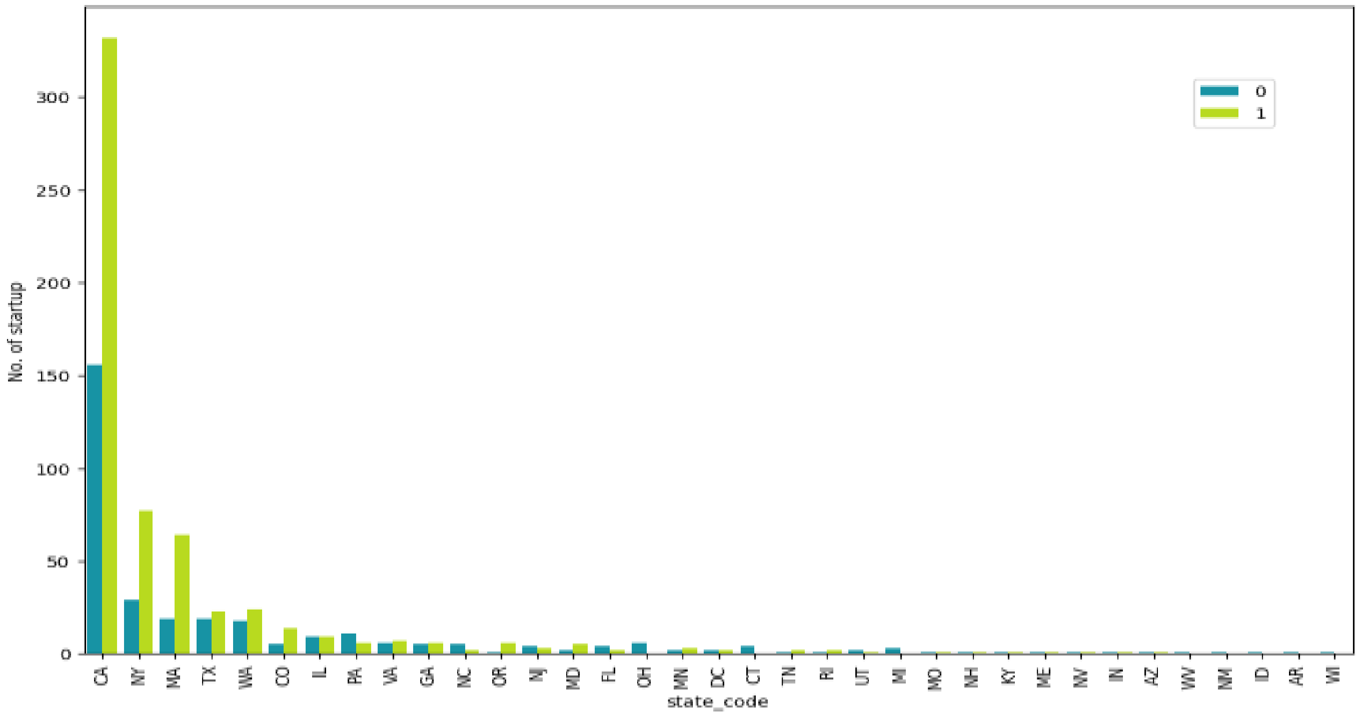


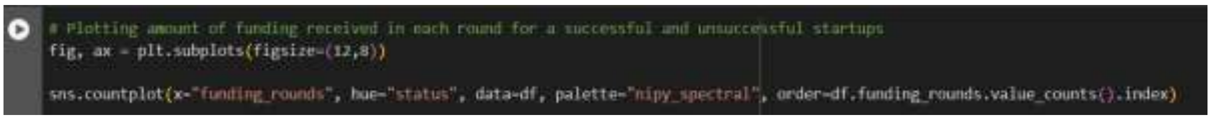

```
[ ]  # Calculating the funding received for each industry
     funding_sorted_category = pd.pivot_table(df,
                   index=['category_code'],
                   values=['funding_total_usd'],
                   aggfunc=['sum']
                   ).reset_index()
     funding_sorted_category.columns = ['category_code', 'funding_total_usd']
     funding_sorted_category = funding_sorted_category.sort_values(['funding_total_usd'], ascending = False)

     fig, ax = plt.subplots(figsize=(15,7))
     plot = sns.barplot(x="category_code", y="funding_total_usd", data=funding_sorted_category,
                     palette="nipy_spectral", ax=ax)
     plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
     plot = ax.set(xlabel="Category", ylabel="Total Funding USD")
```



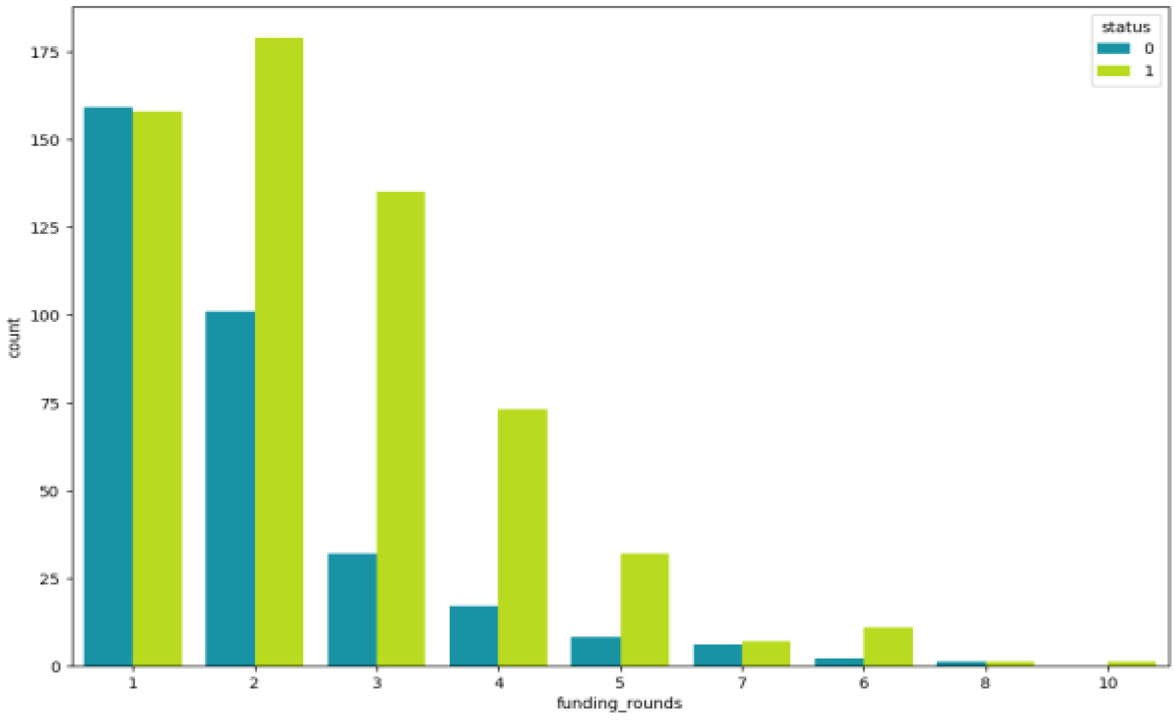


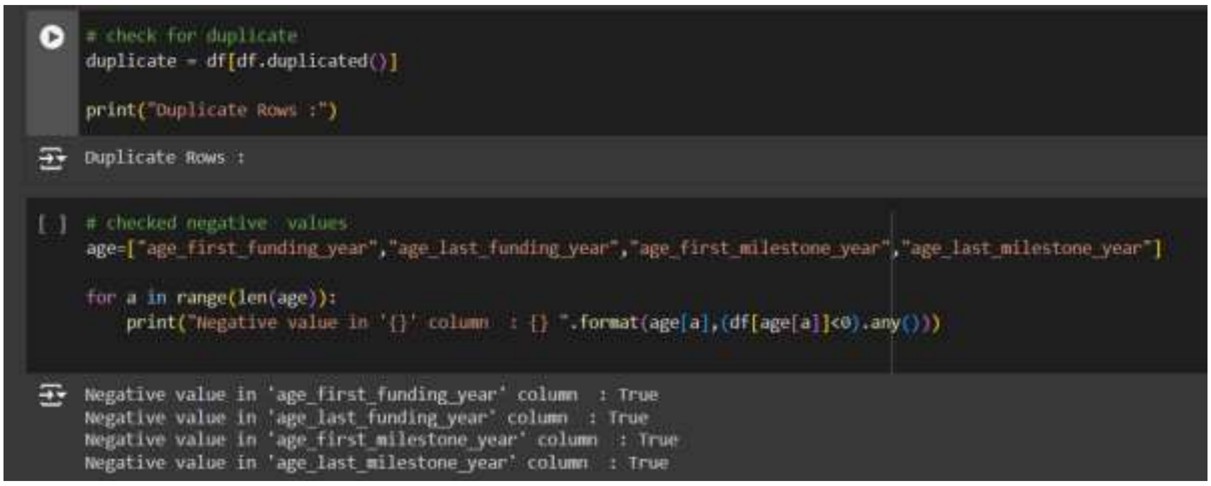
```
[ ]  # Plotting for number of startup in each industry
     fig, ax = plt.subplots(figsize=(12,8))

     plot = sns.countplot(x="state_code", hue="status", data=df, palette="nipy_spectral",
                          order=df.state_code.value_counts().index)

     plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
     plot = ax.set(xlabel="state_code", ylabel="No. of startup")
     plt.legend(bbox_to_anchor=(0.945, 0.90))
```





```
# Plotting amount of funding received in each round for a successful and unsuccessful startups
fig, ax = plt.subplots(figsize=(12,8))

sns.countplot(x="funding_rounds", hue="status", data=df, palette="nipy_spectral", order=df.funding_rounds.value_counts().index)
```



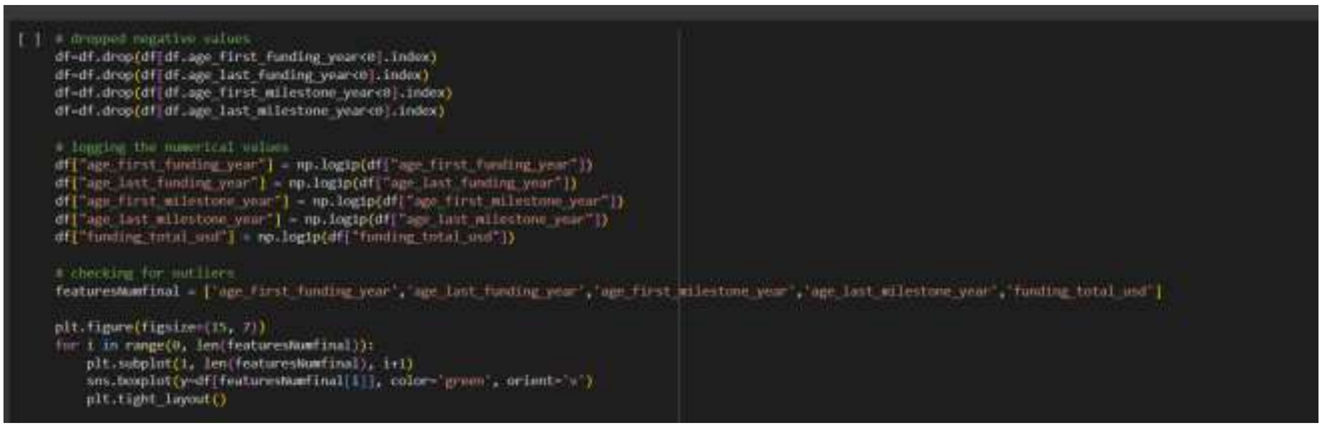


```
# check for duplicate
duplicate = df[df.duplicated()]

print("Duplicate Rows :")
```

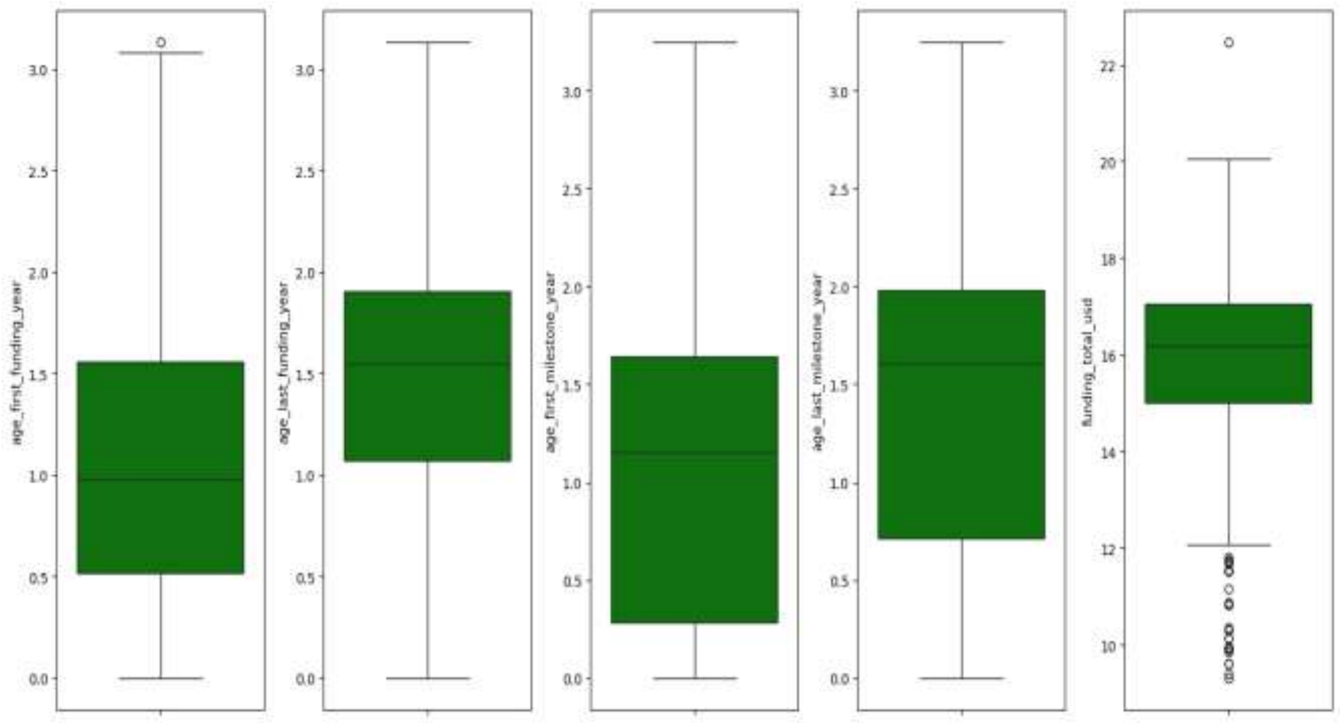
```
Duplicate Rows :
```

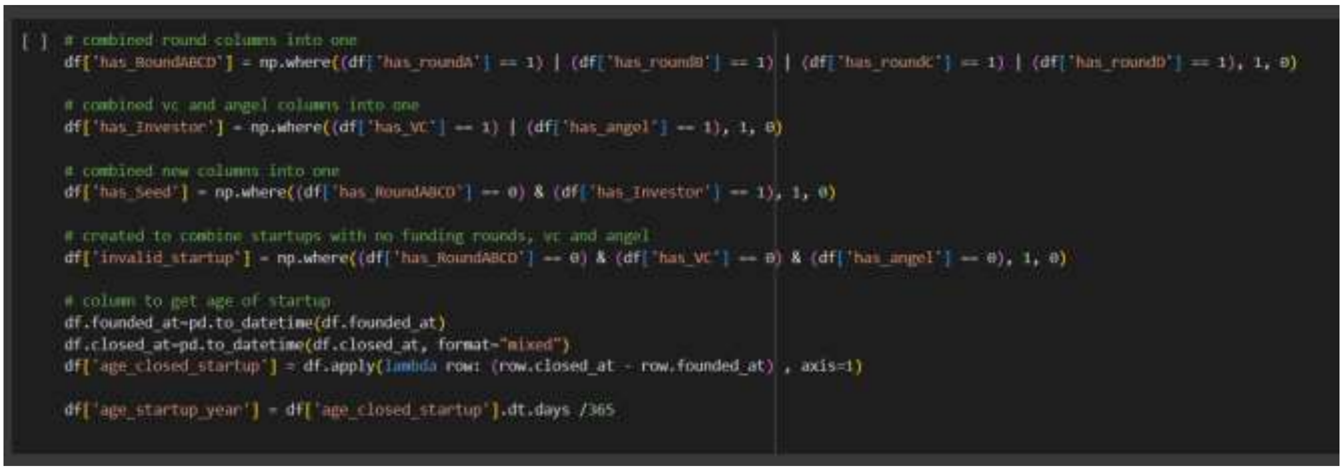
```
# checked negative  values
age=["age_first_funding_year","age_last_funding_year","age_first_milestone_year","age_last_milestone_year"]

for a in range(len(age)):
    print("Negative value in '{}' column  : {} ".format(age[a],(df[age[a]]<0).any()))
```

```
Negative value in 'age_first_funding_year' column  : True
Negative value in 'age_last_funding_year' column  : True
Negative value in 'age_first_milestone_year' column  : True
Negative value in 'age_last_milestone_year' column  : True
```

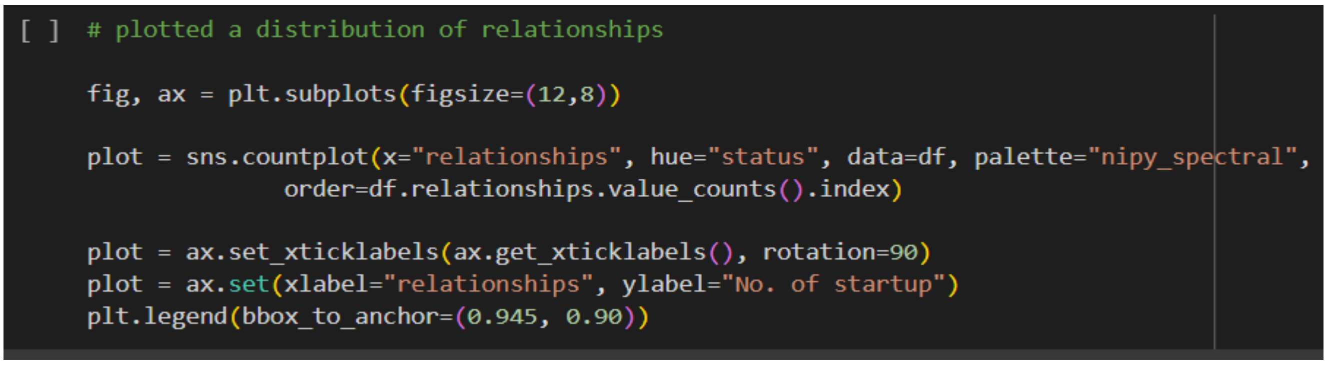


```
# dropped negative values
df=df.drop(df[df.age_first_funding_year<0].index)
df=df.drop(df[df.age_last_funding_year<0].index)
df=df.drop(df[df.age_first_milestone_year<0].index)
df=df.drop(df[df.age_last_milestone_year<0].index)

# logging the numerical values
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"])

# checking for outliers
featuresNumfinal = ['age_first_funding_year','age_last_funding_year','age_first_milestone_year','age_last_milestone_year','funding_total_usd']

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

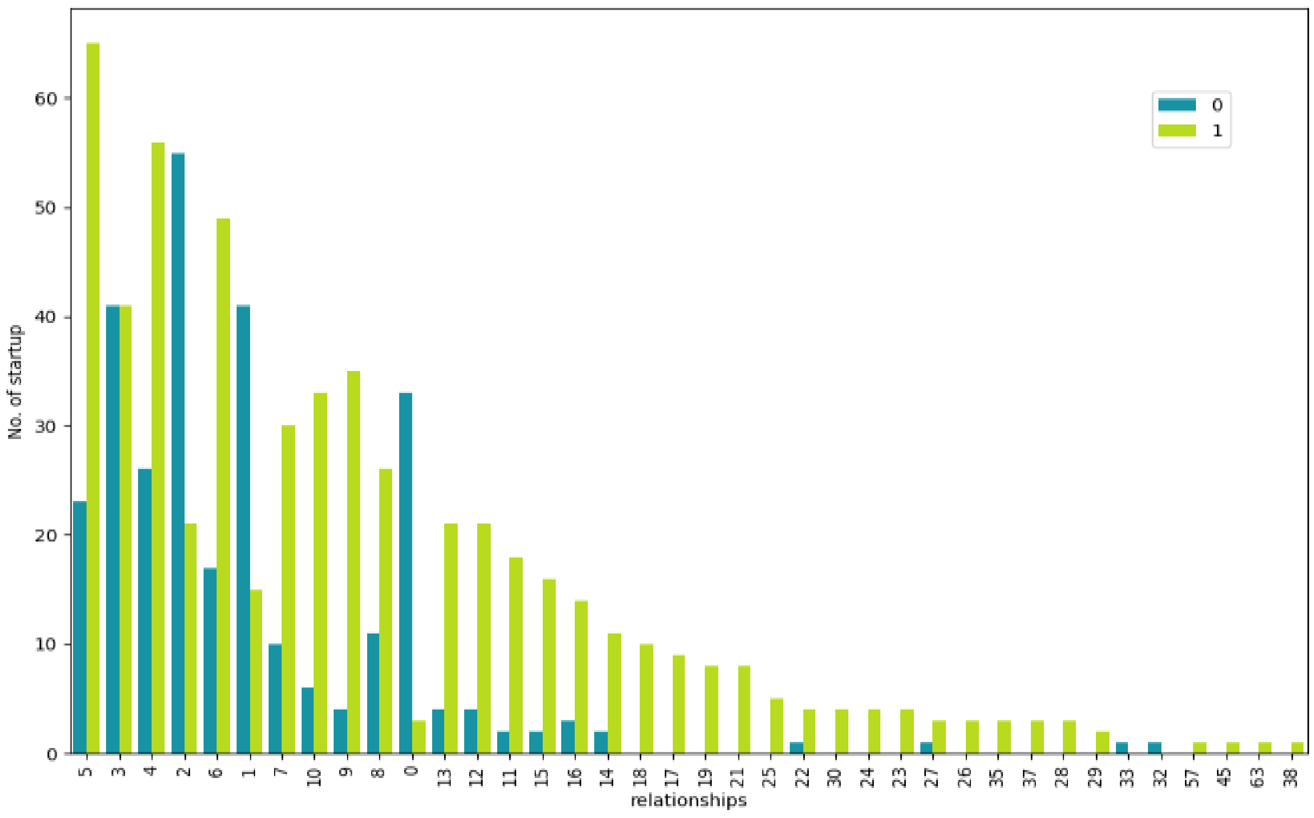


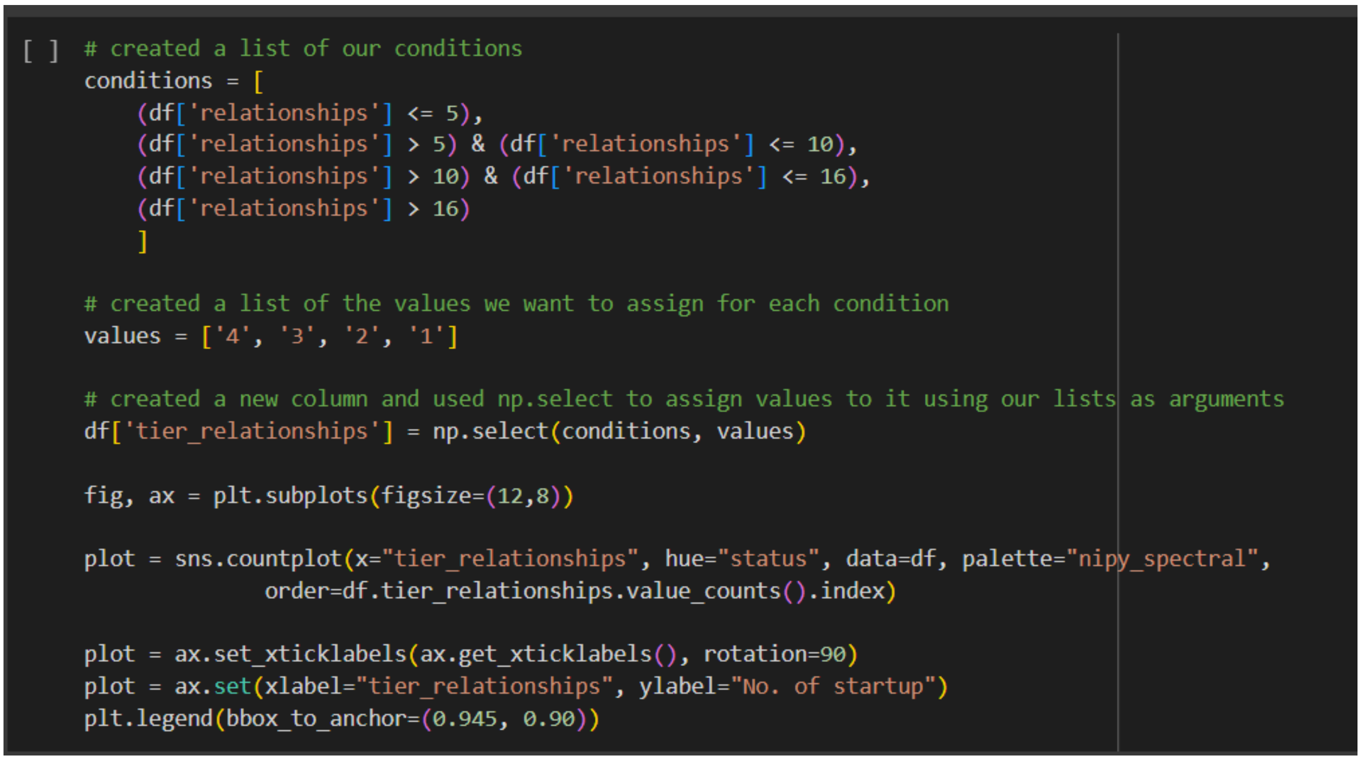


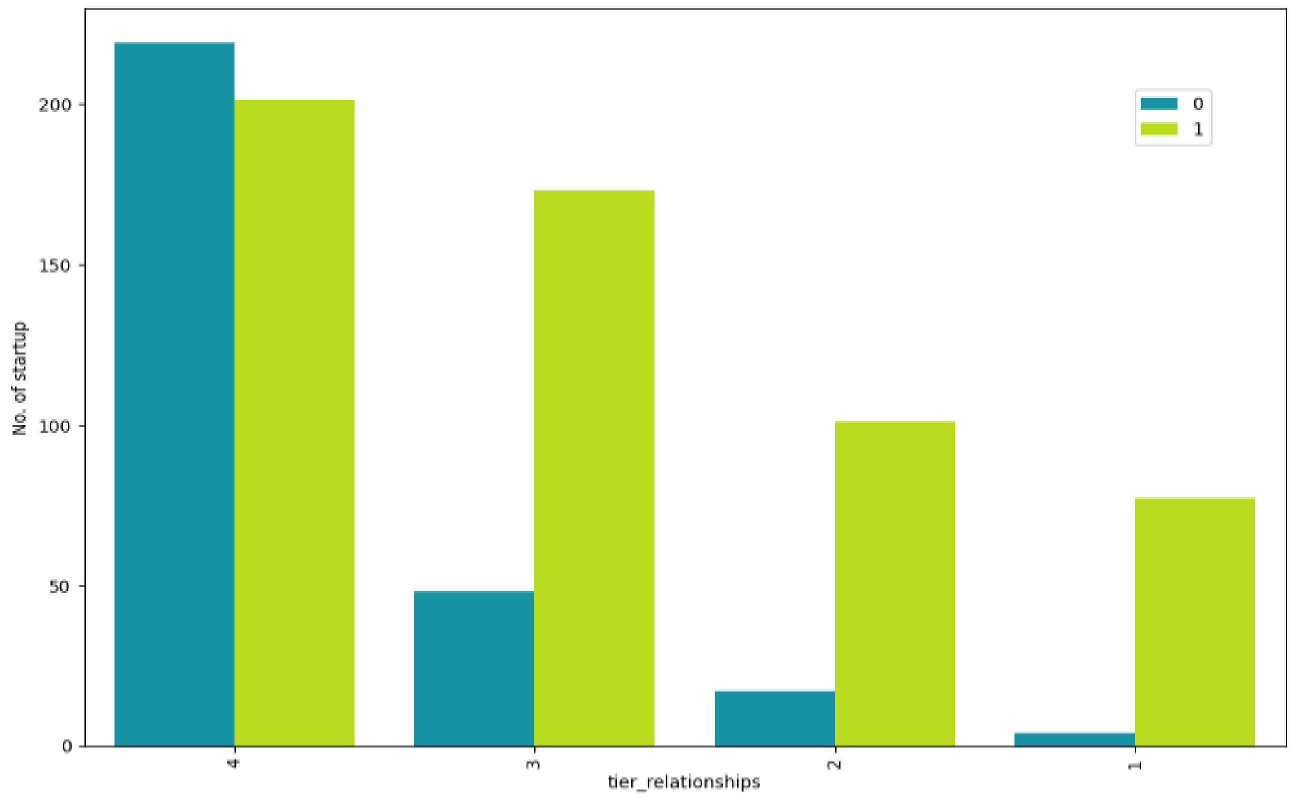
```
[ ] # combined round columns into one
    df['has_RoundABCD'] = np.where((df['has_roundA'] == 1) | (df['has_roundB'] == 1) | (df['has_roundC'] == 1) | (df['has_roundD'] == 1), 1, 0)

    # combined vc and angel columns into one
    df['has_Investor'] = np.where((df['has_VC'] == 1) | (df['has_angel'] == 1), 1, 0)

    # combined new columns into one
    df['has_Seed'] = np.where((df['has_RoundABCD'] == 0) & (df['has_Investor'] == 1), 1, 0)

    # created to combine startups with no funding rounds, vc and angel
    df['invalid_startup'] = np.where((df['has_RoundABCD'] == 0) & (df['has_VC'] == 0) & (df['has_angel'] == 0), 1, 0)

    # column to get age of startup
    df.founded_at=pd.to_datetime(df.founded_at)
    df.closed_at=pd.to_datetime(df.closed_at, format="mixed")
    df['age_closed_startup'] = df.apply(lambda row: (row.closed_at - row.founded_at) , axis=1)

    df['age_startup_year'] = df['age_closed_startup'].dt.days /365
```

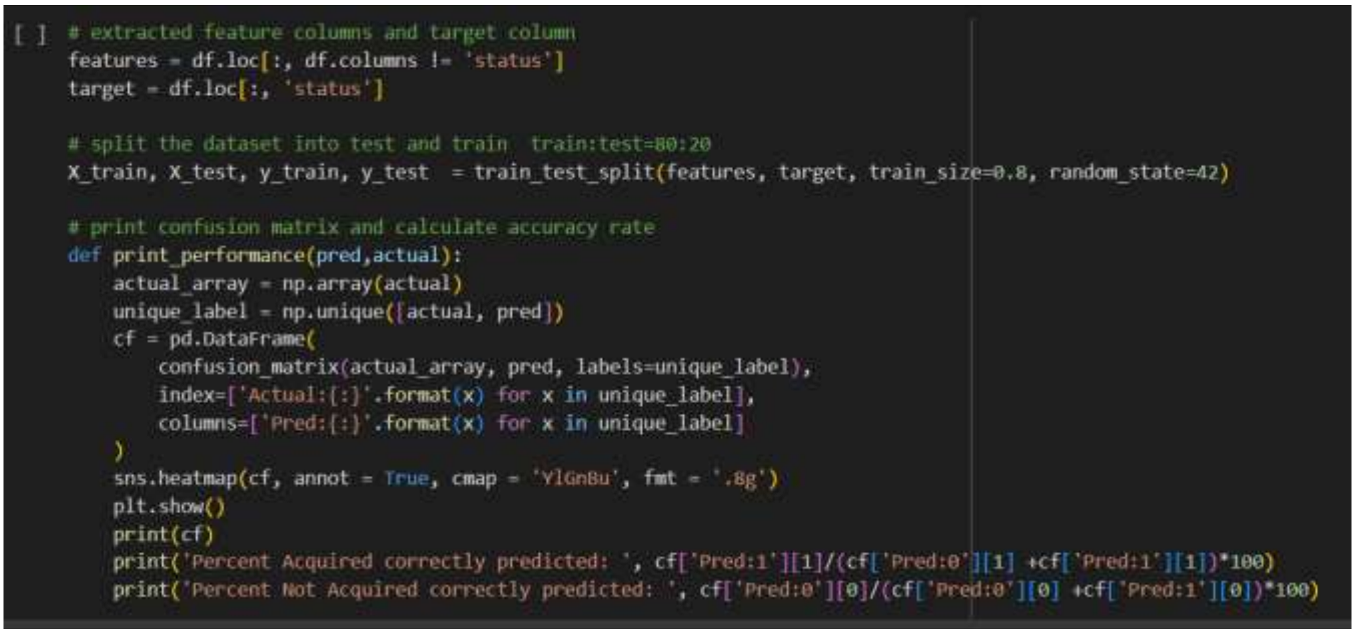


```
[ ] # plotted a distribution of relationships

    fig, ax = plt.subplots(figsize=(12,8))

    plot = sns.countplot(x="relationships", hue="status", data=df, palette="nipy_spectral",
                   order=df.relationships.value_counts().index)

    plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    plot = ax.set(xlabel="relationships", ylabel="No. of startup")
    plt.legend(bbox_to_anchor=(0.945, 0.90))
```

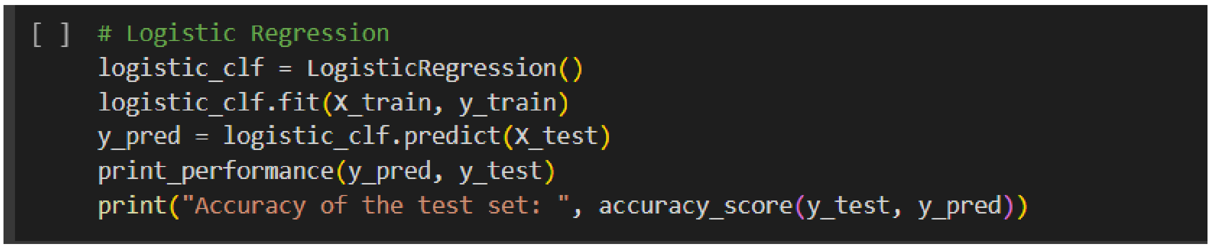



```
[ ] # created a list of our conditions
    conditions = [
        (df['relationships'] <= 5),
        (df['relationships'] > 5) & (df['relationships'] <= 10),
        (df['relationships'] > 10) & (df['relationships'] <= 16),
        (df['relationships'] > 16)
        ]

    # created a list of the values we want to assign for each condition
    values = ['4', '3', '2', '1']

    # created a new column and used np.select to assign values to it using our lists as arguments
    df['tier_relationships'] = np.select(conditions, values)

    fig, ax = plt.subplots(figsize=(12,8))

    plot = sns.countplot(x="tier_relationships", hue="status", data=df, palette="nipy_spectral",
                 order=df.tier_relationships.value_counts().index)

    plot = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    plot = ax.set(xlabel="tier_relationships", ylabel="No. of startup")
    plt.legend(bbox_to_anchor=(0.945, 0.90))
```

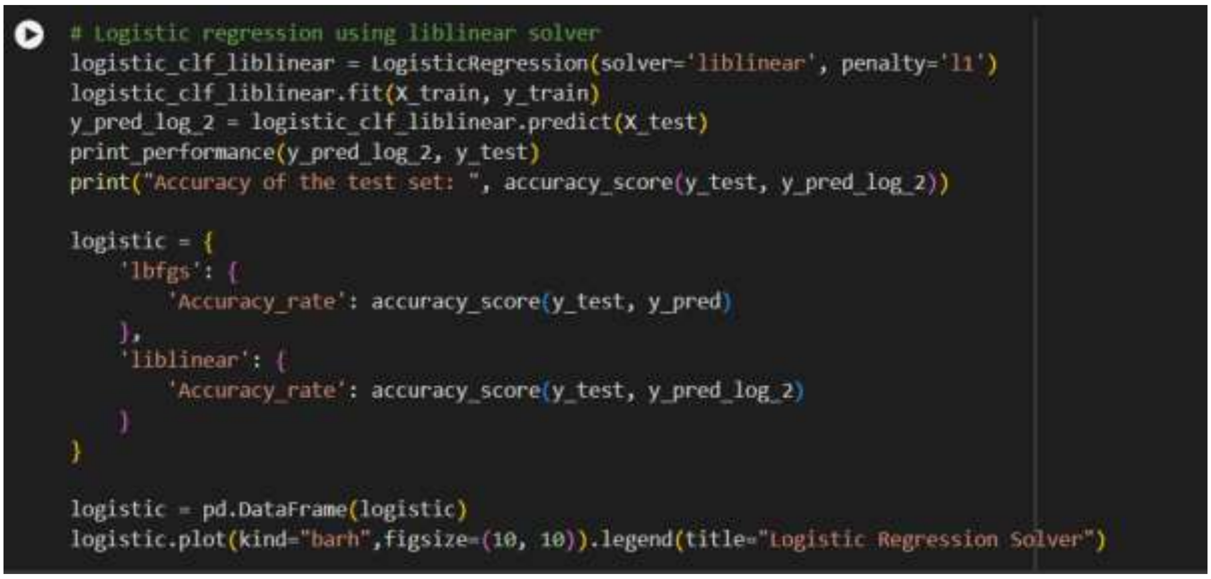




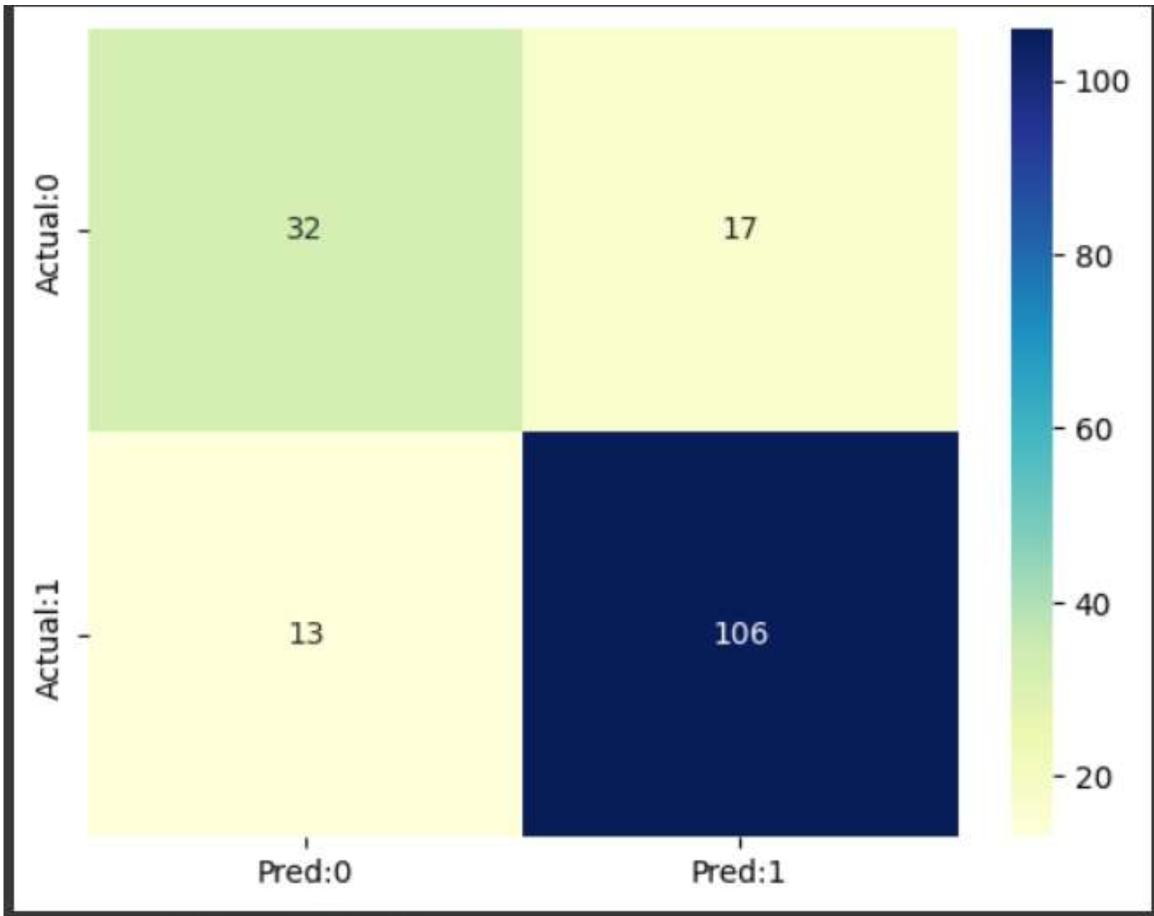
```
[ ] # extracted feature columns and target column
    features = df.loc[:, df.columns != 'status']
    target = df.loc[:, 'status']

    # split the dataset into test and train  train:test=80:20
    X_train, X_test, y_train, y_test  = train_test_split(features, target, train_size=0.8, random_state=42)

    # print confusion matrix and calculate accuracy rate
    def print_performance(pred,actual):
        actual_array = np.array(actual)
        unique_label = np.unique([actual, pred])
        cf = pd.DataFrame(
            confusion_matrix(actual_array, pred, labels=unique_label),
            index=['Actual:{:}'.format(x) for x in unique_label],
            columns=['Pred:{:}'.format(x) for x in unique_label]
        )
        sns.heatmap(cf, annot = True, cmap = 'YlGnBu', fmt = '.8g')
        plt.show()
        print(cf)
        print('Percent Acquired correctly predicted: ', cf['Pred:1'][1]/(cf['Pred:0'][1] +cf['Pred:1'][1])*100)
        print('Percent Not Acquired correctly predicted: ', cf['Pred:0'][0]/(cf['Pred:0'][0] +cf['Pred:1'][0])*100)
```



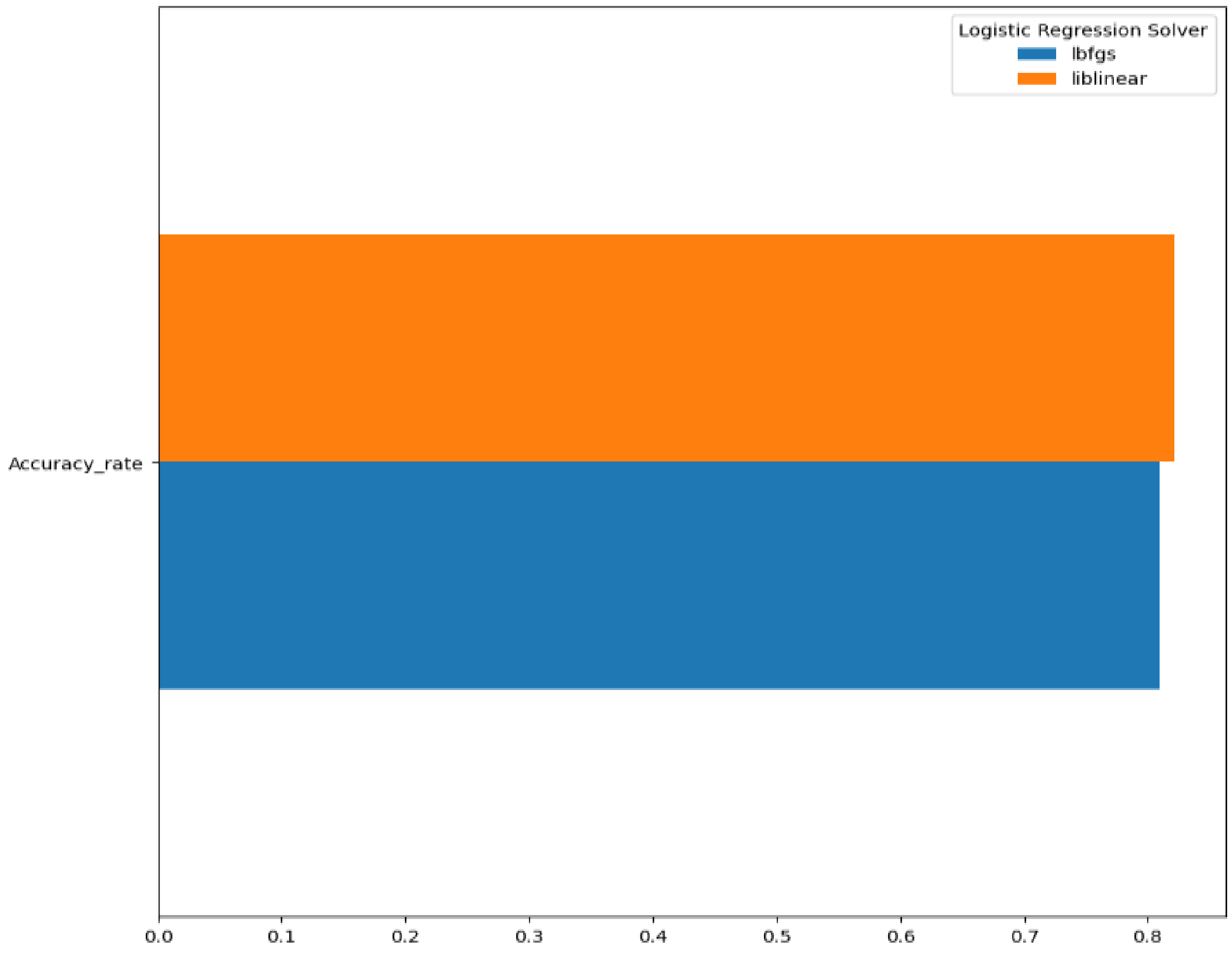
```
[ ] # Logistic Regression
    logistic_clf = LogisticRegression()
    logistic_clf.fit(X_train, y_train)
    y_pred = logistic_clf.predict(X_test)
    print_performance(y_pred, y_test)
    print("Accuracy of the test set: ", accuracy_score(y_test, y_pred))
```

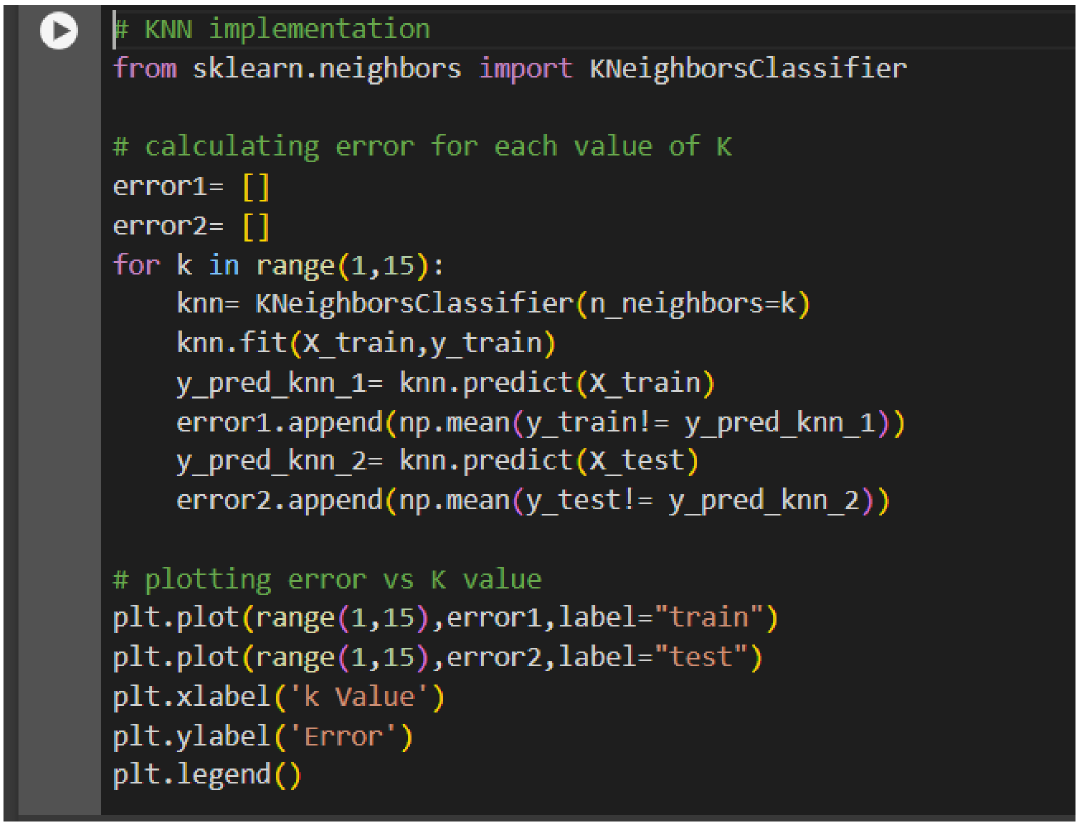


```
# Logistic regression using liblinear solver
logistic_clf_liblinear = LogisticRegression(solver='liblinear', penalty='l1')
logistic_clf_liblinear.fit(X_train, y_train)
y_pred_log_2 = logistic_clf_liblinear.predict(X_test)
print_performance(y_pred_log_2, y_test)
print("Accuracy of the test set: ", accuracy_score(y_test, y_pred_log_2))

logistic = {
    'lbfgs': {
        'Accuracy_rate': accuracy_score(y_test, y_pred)
    },
    'liblinear': {
        'Accuracy_rate': accuracy_score(y_test, y_pred_log_2)
    }
}

logistic = pd.DataFrame(logistic)
logistic.plot(kind="barh",figsize=(10, 10)).legend(title="Logistic Regression Solver")
```

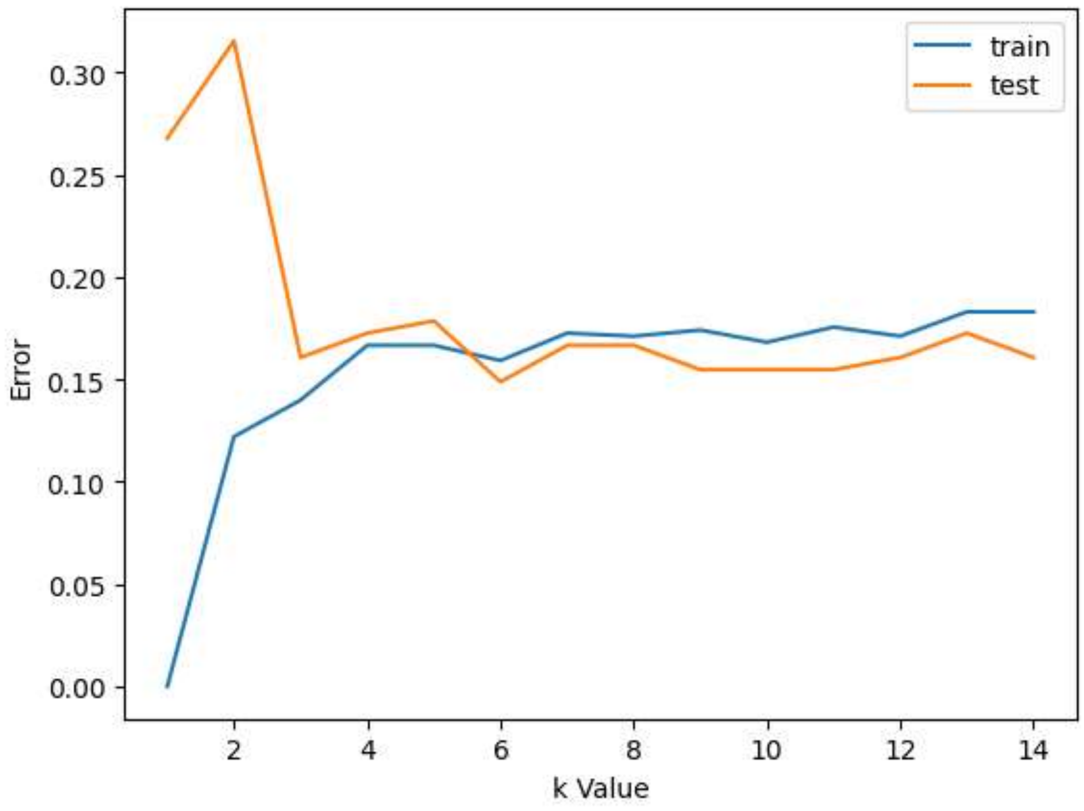


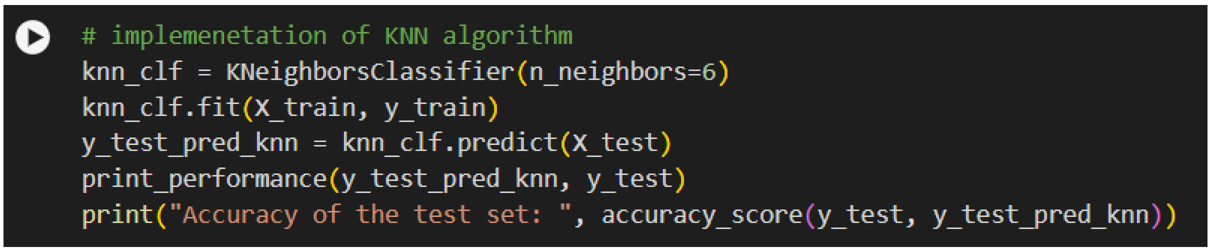
```
         Pred:0  Pred:1
Actual:0      32      17
Actual:1      13     106
Percent Acquired correctly predicted:  89.07563025210085
Percent Not Acquired correctly predicted:  65.3061224489796
Accuracy of the test set:  0.8214285714285714
<matplotlib.legend.Legend at 0x7cb81fae33d0>
```



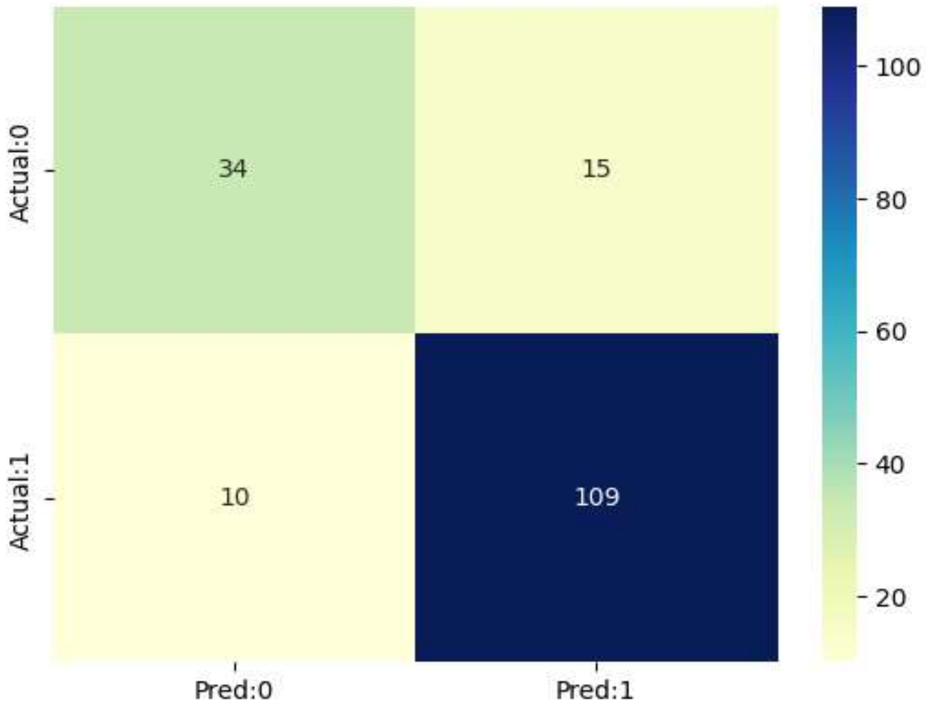


```
# KNN implementation
from sklearn.neighbors import KNeighborsClassifier

# calculating error for each value of K
error1= []
error2= []
for k in range(1,15):
    knn= KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    y_pred_knn_1= knn.predict(X_train)
    error1.append(np.mean(y_train!= y_pred_knn_1))
    y_pred_knn_2= knn.predict(X_test)
    error2.append(np.mean(y_test!= y_pred_knn_2))

# plotting error vs K value
plt.plot(range(1,15),error1,label="train")
plt.plot(range(1,15),error2,label="test")
plt.xlabel('k Value')
plt.ylabel('Error')
plt.legend()
```



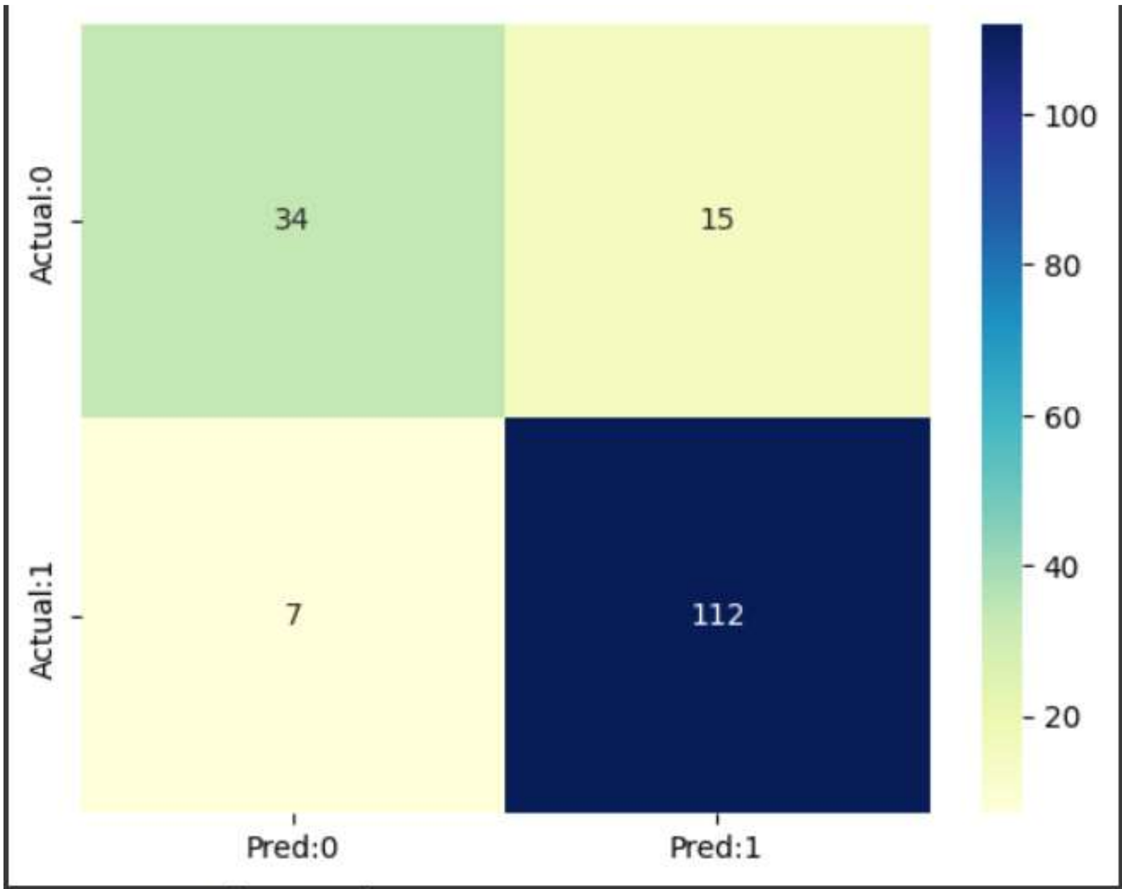


```
# implemenetation of KNN algorithm
knn_clf = KNeighborsClassifier(n_neighbors=6)
knn_clf.fit(X_train, y_train)
y_test_pred_knn = knn_clf.predict(X_test)
print_performance(y_test_pred_knn, y_test)
print("Accuracy of the test set: ", accuracy_score(y_test, y_test_pred_knn))
```

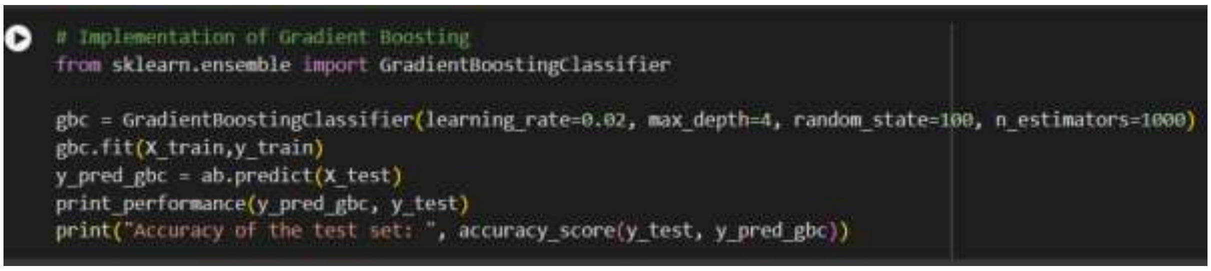


```
              Pred:0   Pred:1
Actual:0          34       15
Actual:1          10      109
Percent Acquired correctly predicted:  91.59663865546219
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8511904761904762
```

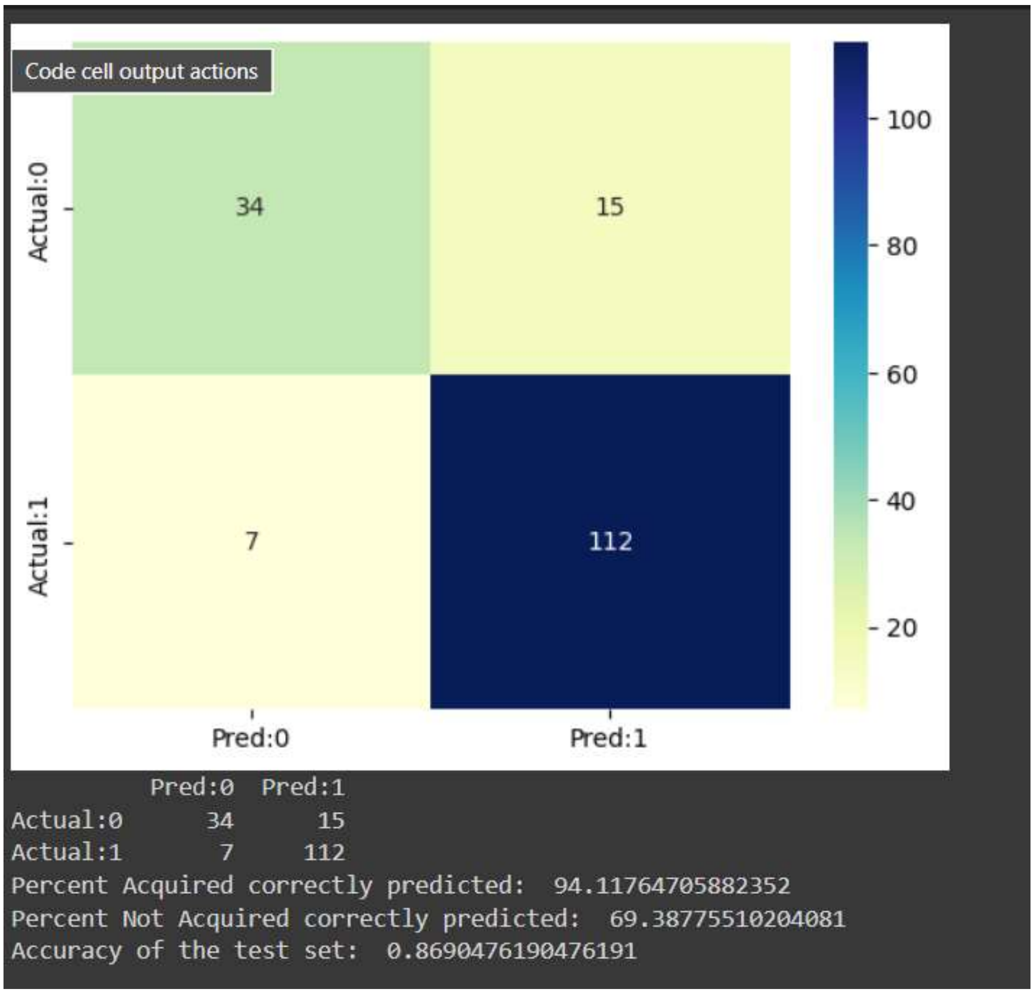
```
[ ]  # Implementaion of Adaptive Boosting
     from sklearn.ensemble import AdaBoostClassifier

     ab = AdaBoostClassifier(random_state=42)
     ab.fit(X_train, y_train)
     y_pred_ada = ab.predict(X_test)
     print_performance(y_pred_ada, y_test)
     print("Accuracy of the test set: ", accuracy_score(y_test, y_pred_ada))
```



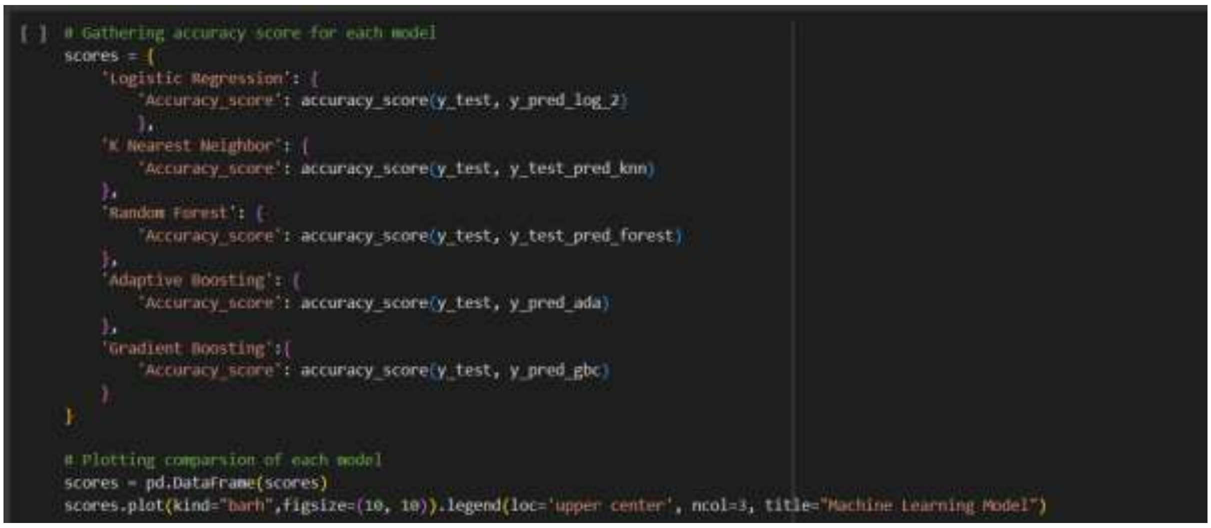
```
              Pred:0   Pred:1
Actual:0          34       15
Actual:1           7      112
Percent Acquired correctly predicted:  94.11764705882352
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8690476190476191
```



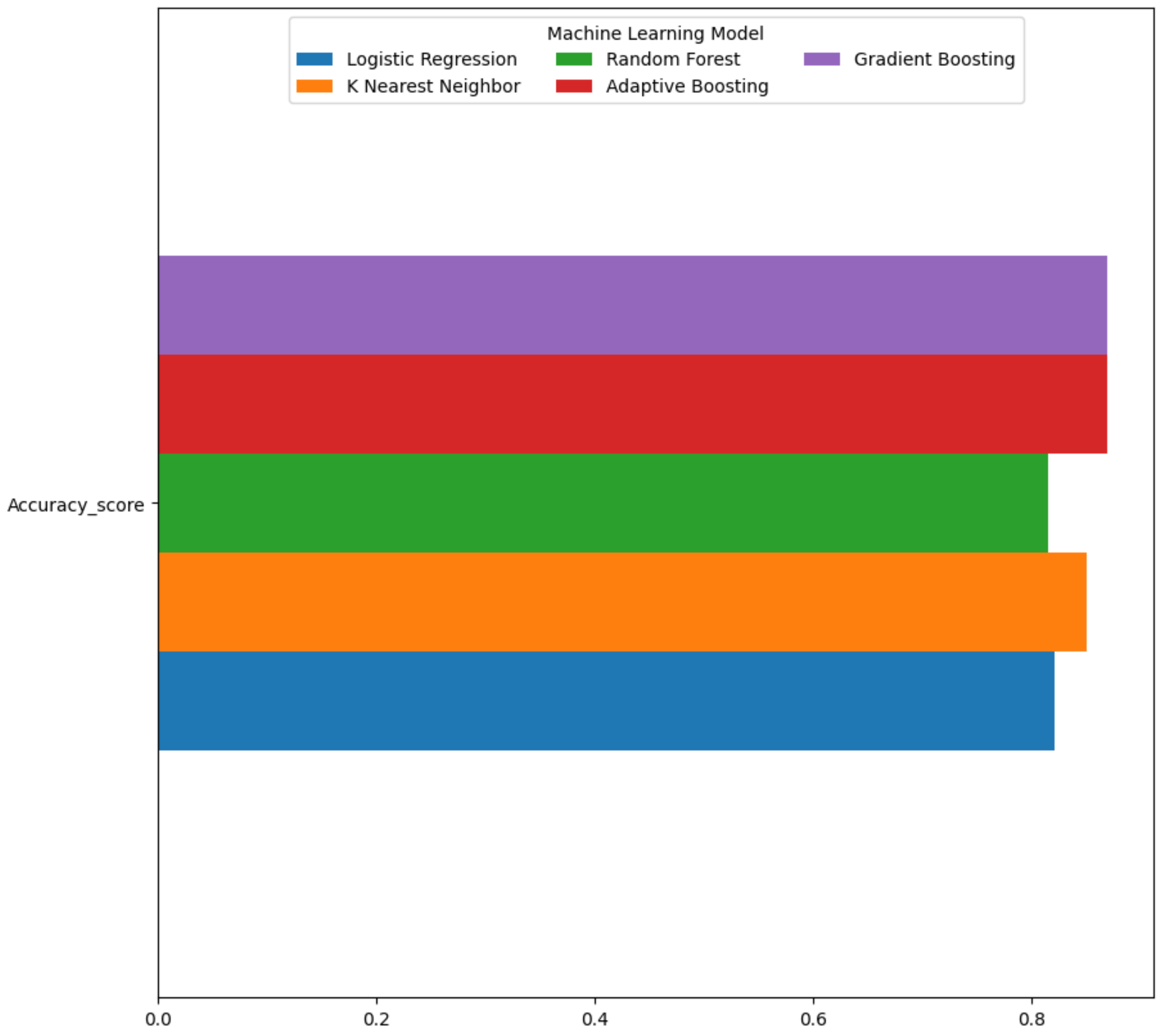
```
# Implementation of Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(learning_rate=0.02, max_depth=4, random_state=100, n_estimators=1000)
gbc.fit(X_train,y_train)
y_pred_gbc = ab.predict(X_test)
print_performance(y_pred_gbc, y_test)
print("Accuracy of the test set: ", accuracy_score(y_test, y_pred_gbc))
```



```
         Pred:0  Pred:1
Actual:0      34      15
Actual:1       7     112
Percent Acquired correctly predicted:  94.11764705882352
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8690476190476191
```



```
[ ] # Gathering accuracy score for each model
    scores = {
        'Logistic Regression': {
            'Accuracy_score': accuracy_score(y_test, y_pred_log_2)
            },
        'K Nearest Neighbor': {
            'Accuracy_score': accuracy_score(y_test, y_test_pred_knn)
        },
        'Random Forest': {
            'Accuracy_score': accuracy_score(y_test, y_test_pred_forest)
        },
        'Adaptive Boosting': {
            'Accuracy_score': accuracy_score(y_test, y_pred_ada)
        },
        'Gradient Boosting':{
            'Accuracy_score': accuracy_score(y_test, y_pred_gbc)
        }
    }

    # Plotting comparsion of each model
    scores = pd.DataFrame(scores)
    scores.plot(kind="barh",figsize=(10, 10)).legend(loc='upper center', ncol=3, title="Machine Learning Model")
```


Machine Learning Model
Logistic Regression
K Nearest Neighbor
Random Forest
Adaptive Boosting
Gradient Boosting
Accuracy_score
0.0
0.2
0.4
0.6
0.8