Prosperity Prognosticator – Machine Learning for Startup Success Prediction

1. Introduction

Project Title: Prosperity prognosticator: machine learning for startup success prediction
Team ID: LTVIP2026TMIDS47450
Team Size: 4

- **Team Leader:** Diviti Sowmya– Project Coordination, Milestone Tracking, Data Collection, Cleaning.
- **Team Member:** Mudipalli Kalpana– Preprocessing pipeline, Model Building, Testing, UI/UX Design.
- **Team Member:** Nimmala Sravanthi– Documentation, Descriptive Statistics, Model Evaluation.
- **Team Member:** Banda Vijay Kumar– Define Problem/Problem Understanding, ML Model Development & Flask Integration.

2. Project Overview

2.1. Purpose

Startups play a major role in innovation and economic growth, but predicting startup success is difficult due to multiple influencing factors such as funding, market conditions, and operational strategies. This project, Prosperity Prognosticator, aims to build a machine learning model to predict startup success using structured startup data. The goal is to provide a fast, accurate, and accessible solution through a web interface that assists investors, entrepreneurs, and analysts in making data-driven decisions.

2.2. Features

• User interface to enter startup details
• Prediction using trained Random Forest machine learning model
• Real-time result display on web interface
• Error handling for invalid inputs
• Instant prediction output: "Successful" or "Failed" startup

3. Architecture

3.1. Frontend

• Developed Using: HTML5, CSS3
• HTML Pages (templates folder):
  - ➢ **index.html:** Form to enter startup information
  - ➢ **results.html:** Displays prediction output

- User Experience:
  - ➢ Clean and minimal interface
  - ➢ Responsive layout
  - ➢ Dynamic result display after prediction

3.2. Backend

- Framework: Python with Flask microframework

Prediction Workflow

- User enters startup details via form
- Input data received by Flask backend
- Data preprocessing applied
- Trained model (random_forest_model.pkl) loaded
- Model predicts startup outcome
- Result displayed on results.html

Model Training

- Training done in Google Colab / Jupiter Notebook
- Dataset cleaned, encoded, and scaled
- Multiple algorithms tested
- Best model saved as .pkl file

Files Used

- app.py – routing, preprocessing, prediction logic

random_forest_model.pkl – trained ML model

3.3. Data Preprocessing

- Missing value handling
- Categorical feature encoding
- Feature scaling (StandardScaler)
- Train-test split (70:30)
- Input converted to model-ready numerical format

3.4. Database

- No database integrated
- Dataset handled locally through CSV/Excel files
- Model reads input and returns prediction instantly

4. Setup Instructions

4.1. Prerequisites

- Python 3.10+
- Flask
- Scikit-learn
- Pandas, NumPy
- Matplotlib, Seaborn
- Jupyter Notebook / Google Colab

4.2. Installation

```
pip install -r requirements.txt
python app.py
```

5. Folder Structure

5.1. Client

```
startup-success-prediction/
│
├── templates/
│   ├── index.html
│   └── result.html
│
├── Startup_Success_Prediction (1). ipynb
├── app.py
├── random_forest_model.pkl
├── startup1.csv
├── .gitignore
```

5.2. File / Directory Roles

- startup1.csv → Dataset used for training and testing the machine learning model.
- templates/ → Contains HTML pages that provide the user interface for the Flask application.
- app.py → Main backend application responsible for routing, data processing, model loading, and prediction generation
- random_forest_model.pkl → Serialized trained Random Forest model used for real-time prediction.
- Startup_Success_Prediction (1). ipynb → Jupyter Notebook used for data preprocessing, exploratory analysis, model training, and evaluation.

6. Running the Application

To run the application:

python app.py

After execution, open a browser and navigate to:

http://127.0.0.1:5000/

The web interface will load, allowing users to enter startup details and obtain predictions.

7. API Document

Route: /predict

- **Method:** POST (via HTML form submission)
- **Input:** Startup-related features entered by the user through the web interface.
- Processing:
  Data preprocessing and formatting
  Model inference using the trained Random Forest classifier
- Output:
  Rendered HTML page showing startup success prediction result.
- **Error Handling:**
  Handles missing or invalid input values gracefully and prevents application crashes.

8. Authentication

Authentication is not implemented in this version since the project is a prototype intended for demonstration and research purposes only.

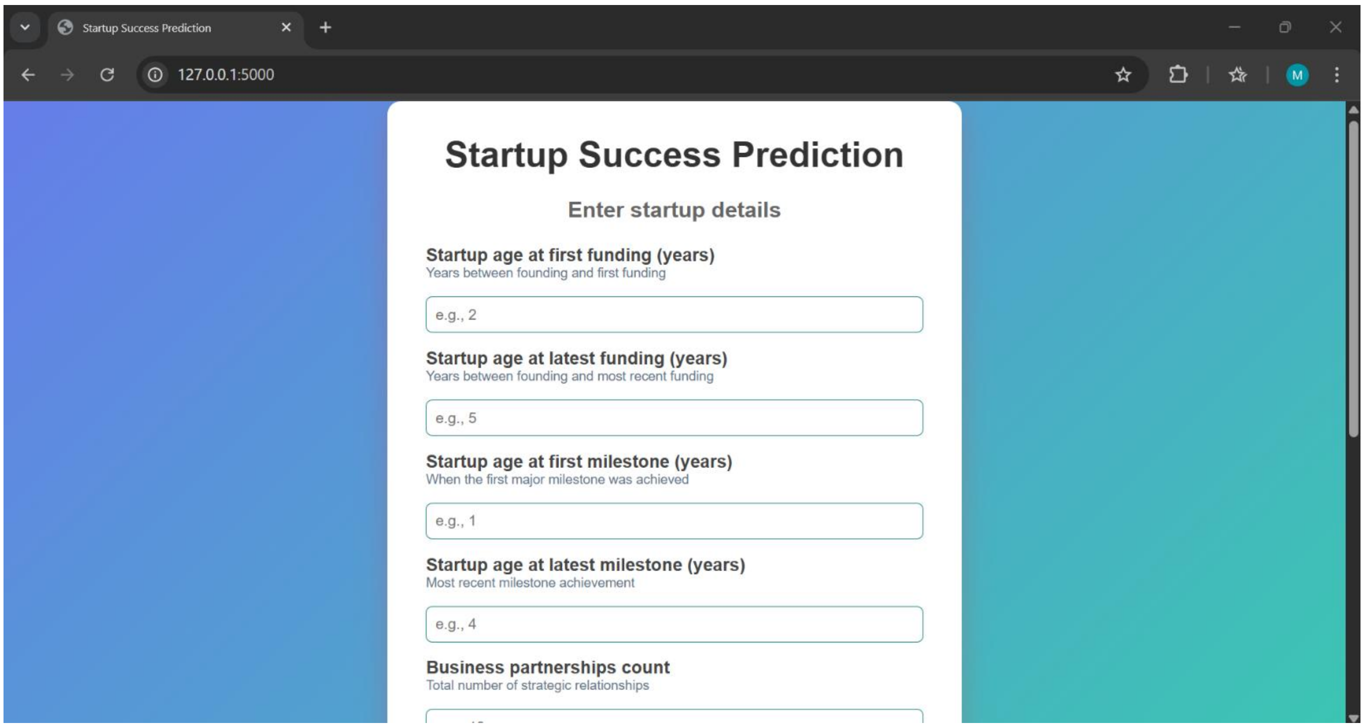
9. User Interface

- ➢ Simple and intuitive form for entering startup attributes.
- ➢ Clean and responsive layout designed for easy understanding.
- ➢ Prediction result displayed clearly after submission.

10. Testing

- Model evaluated using train–test split methodology.
- Evaluation metrics used:
  - ➢ Accuracy
  - ➢ Precision
  - ➢ Recall
  - ➢ F1-Score
  - ➢ Confusion Matrix
- Manual testing performed using sample startup data to validate prediction workflow and UI integration.

11. Screenshots or Demo



Startup Success Prediction

Enter startup details

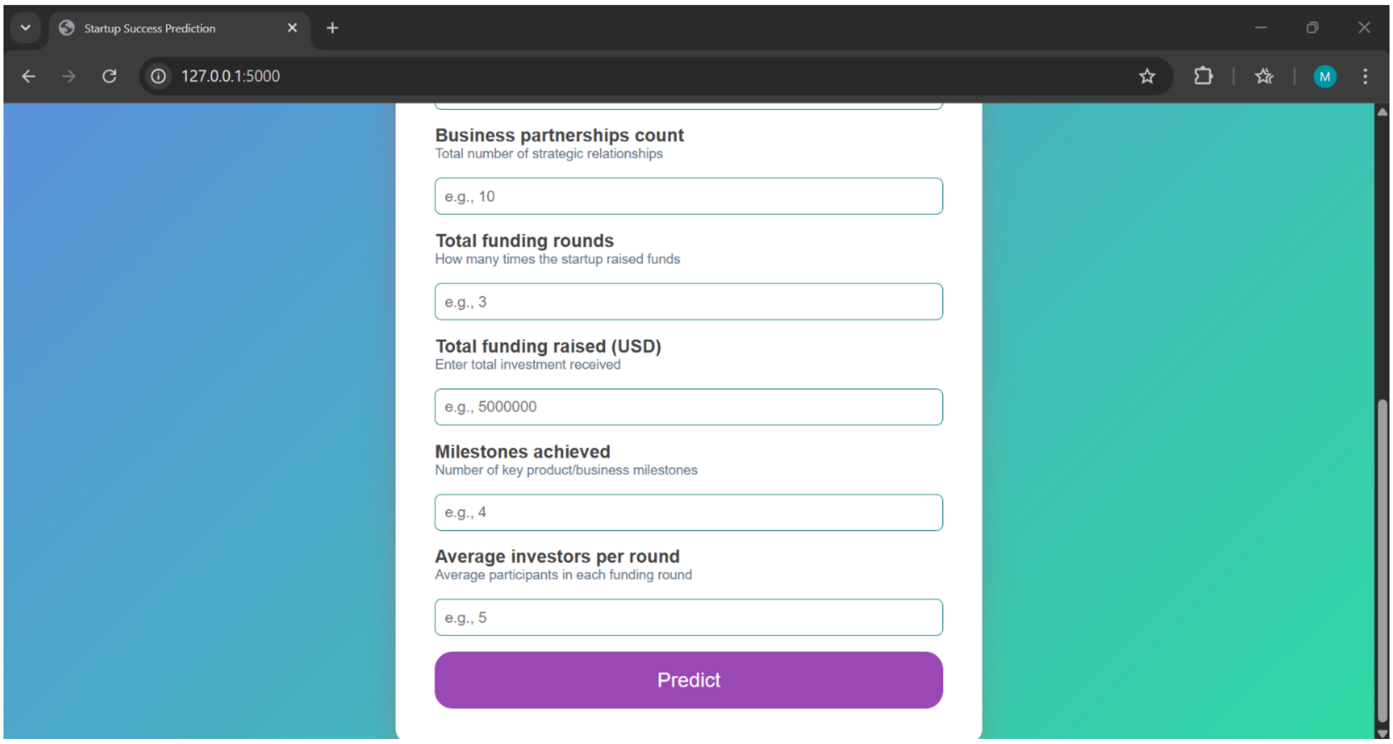
Startup age at first funding (years)
Years between founding and first funding
e.g., 2

Startup age at latest funding (years)
Years between founding and most recent funding
e.g., 5

Startup age at first milestone (years)
When the first major milestone was achieved
e.g., 1

Startup age at latest milestone (years)
Most recent milestone achievement
e.g., 4

Business partnerships count
Total number of strategic relationships



Business partnerships count
Total number of strategic relationships
e.g., 10

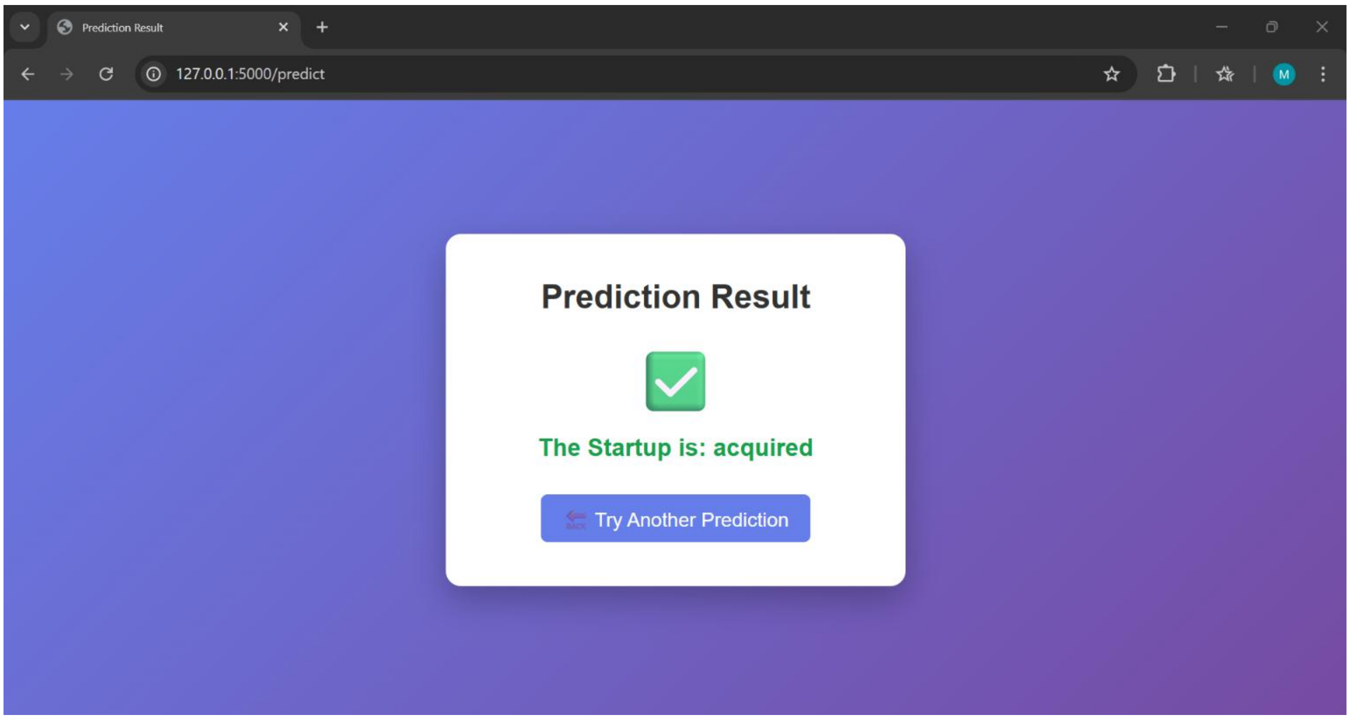
Total funding rounds
How many times the startup raised funds
e.g., 3

Total funding raised (USD)
Enter total investment received
e.g., 5000000

Milestones achieved
Number of key product/business milestones
e.g., 4

Average investors per round
Average participants in each funding round
e.g., 5

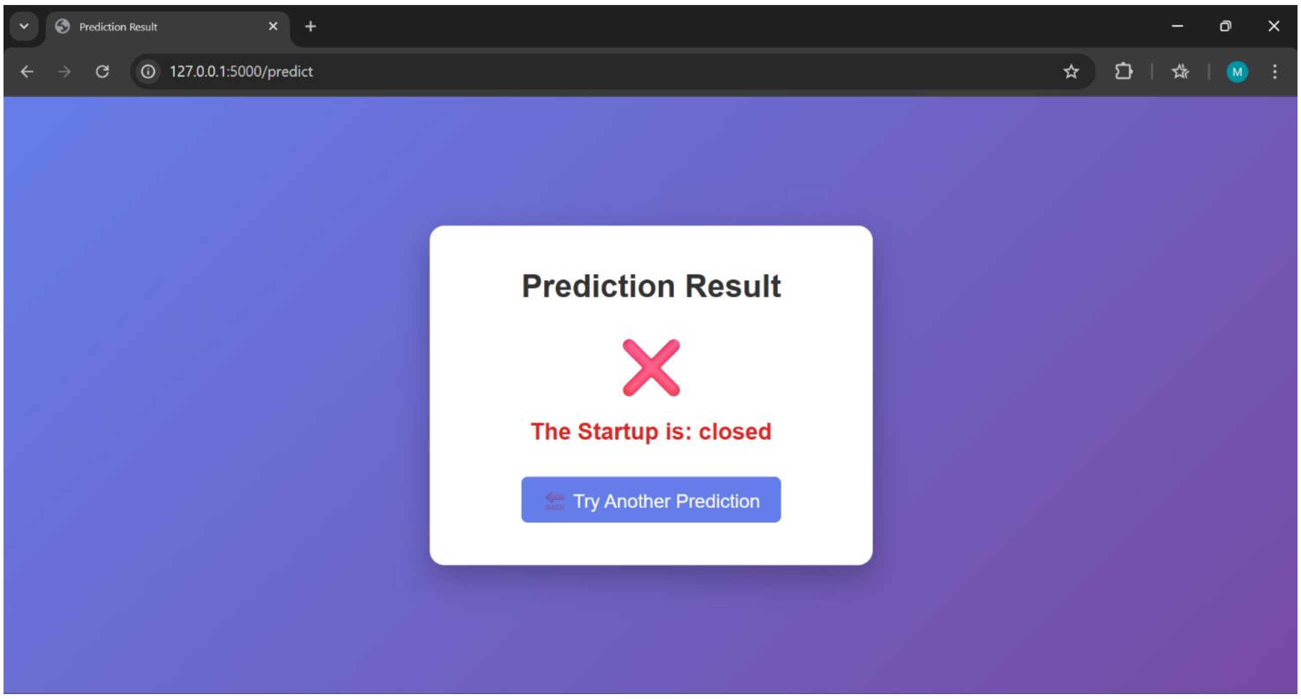
Predict



Prediction Result

The Startup is: acquired

Try Another Prediction



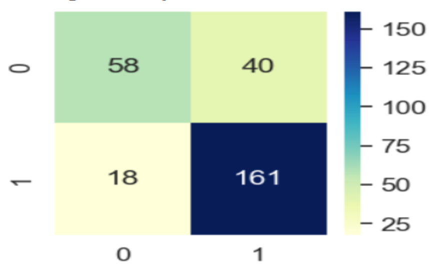
Prediction Result

The Startup is: closed

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        rf = RandomForestClassifier()

        rf.fit(X_train._get_numeric_data(),y_train)


        y_pred_rf = rf.predict(X_test._get_numeric_data())

        print("Training Accuracy :", rf.score(X_train._get_numeric_data(), y_train))
        print("Testing Accuracy :", rf.score(X_test._get_numeric_data(), y_test))

        cm = confusion_matrix(y_test, y_pred_rf)
        plt.rcParams['figure.figsize'] = (3, 3)
        sns.heatmap(cm, annot = True, cmap = 'YlGnBu', fmt = '.8g')
        plt.show()

        cr = classification_report(y_test, y_pred_rf)
        print(cr)


        print("-----------------------------------------")

        false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,y_pred_rf)
        roc_auc = auc(false_positive_rate, true_positive_rate)
        print("ROC Curves              =",roc_auc)

        precision, recall, thresholds = precision_recall_curve(y_test, y_pred_rf)
        f1 = f1_score(y_test, y_pred_rf)
        Precision_Recall_rfs = auc(recall, precision)
        print("Precision-Recall Curves =",Precision_Recall_rfs)
```

```
Training Accuracy : 1.0
Testing Accuracy : 0.7906137184115524
```

```
Training Accuracy : 1.0
Testing Accuracy : 0.7906137184115524
```



```
              precision    recall  f1-score   support

           0       0.76      0.59      0.67        98
           1       0.80      0.90      0.85       179

    accuracy                           0.79       277
   macro avg       0.78      0.75      0.76       277
weighted avg       0.79      0.79      0.78       277

-----------------------------------------
ROC Curves              = 0.7456390377380002
Precision-Recall Curves = 0.8827091575581143
```

- **Demo Link: videodemolink**

12. Known Issues

- ➢ Model performance depends on dataset quality and size.
- ➢ Slight overfitting observed due to high training accuracy compared to testing accuracy.
- ➢ No database integration for storing user predictions or historical results.

13. Future Enhancements

- Expand dataset size for improved generalization.
- Integrate database for logging predictions and analytics.
- Add user authentication and downloadable reports.
- Deploy application on cloud platforms for public access.
- Experiment with advanced models such as XGBoost and Neural Networks for improved performance.