```python
# ================================
# Prosperity Prognosticator
# Single Colab Run Code
# ================================

# 1 Install required libraries
!pip install flask joblib scikit-learn pandas numpy matplotlib seaborn

# 2 Import libraries
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# 3 Upload CSV file
from google.colab import files
uploaded = files.upload()

# Get uploaded filename
filename = list(uploaded.keys())[0]

# 4 Load dataset
# Use pd.read_excel for .xlsx files
if filename.endswith('.xlsx'):
    data = pd.read_excel(filename)
elif filename.endswith('.csv'):
    data = pd.read_csv(filename)
else:
    raise ValueError("Unsupported file format. Please upload a .csv or
.xlsx file.")

print("Dataset Shape:", data.shape)
data.head()

# 5 Basic EDA
print("\nMissing Values:\n", data.isnull().sum())
print("\nStatistical Summary:\n", data.describe())

# Drop non-numeric columns for correlation calculation
numeric_data = data.select_dtypes(include=[np.number])

# Correlation Heatmap
plt.figure(figsize=(8,6))
```

```
sns.heatmap(numeric_data.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation")
plt.show()

# 6 Feature & Target split
# Identify columns to drop (IDs, text, dates that aren't processed,
and redundant/unnamed)
cols_to_drop = [
    'Unnamed: 0', 'id', 'object_id', 'name', 'zip_code', 'city',
    'state_code', 'state_code.1', 'category_code', 'status',
    'founded_at', 'closed_at', 'first_funding_at', 'last_funding_at',
    'Unnamed: 6'
]
# Filter out columns that might not exist in the dataframe after
initial cleaning or if the dataset changes.
cols_to_drop = [col for col in cols_to_drop if col in data.columns]

X = data.drop(columns=cols_to_drop + ["labels"], errors='ignore')
y = data["labels"]

# 7 Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 8 Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 9 Random Forest + GridSearchCV
param_grid = {
    "n_estimators": [100, 200],
    "max_depth": [None, 10, 20],
    "min_samples_split": [2, 5],
    "min_samples_leaf": [1, 2],
    "bootstrap": [True, False]
}

rf = RandomForestClassifier(random_state=42)

grid = GridSearchCV(
    rf,
    param_grid,
    cv=5,
    scoring="accuracy",
    n_jobs=-1
)

grid.fit(X_train_scaled, y_train)
```

```
best_model = grid.best_estimator_

print("\nBest Parameters Found:")
print(grid.best_params_)

# □ Model Evaluation
train_pred = best_model.predict(X_train_scaled)
test_pred = best_model.predict(X_test_scaled)

train_acc = accuracy_score(y_train, train_pred)
test_acc = accuracy_score(y_test, test_pred)

print("\nTraining Accuracy:", train_acc)
print("Testing Accuracy:", test_acc)

print("\nClassification Report:\n")
print(classification_report(y_test, test_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, test_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

#111 Save Model & Scaler
joblib.dump(best_model, "random_forest_model.pkl")
joblib.dump(scaler, "scaler.pkl")

print("\nModel & Scaler Saved Successfully!")

#212 Download Saved Files
files.download("random_forest_model.pkl")
files.download("scaler.pkl")

#313 Generate Flask App Code (Optional Deployment)
flask_code = """
from flask import Flask, render_template, request
import numpy as np
import joblib

app = Flask(__name__)

model = joblib.load("random_forest_model.pkl")
scaler = joblib.load("scaler.pkl")

@app.route("/")
def home():
    return "Startup Success Prediction App"
```

```
@app.route("/predict", methods=["POST"])
def predict():
    data = [float(x) for x in request.form.values()]
    final_data = scaler.transform([data])
    prediction = model.predict(final_data)[0]
    result = "Acquired / Successful" if prediction == 1 else "Closed /
Failed"
    return result

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

with open("app.py", "w") as f:
    f.write(flask_code)

files.download("app.py")

print("\nFlask app.py file generated!")
```

```
Requirement already satisfied: flask in
/usr/local/lib/python3.12/dist-packages (3.1.2)
Requirement already satisfied: joblib in
/usr/local/lib/python3.12/dist-packages (1.5.3)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: pandas in
/usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in
/usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: blinker>=1.9.0 in
/usr/local/lib/python3.12/dist-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in
/usr/local/lib/python3.12/dist-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in
/usr/local/lib/python3.12/dist-packages (from flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in
/usr/local/lib/python3.12/dist-packages (from flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in
/usr/local/lib/python3.12/dist-packages (from flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from flask) (3.1.5)
Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in
```

```
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (26.0)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (3.3.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
```

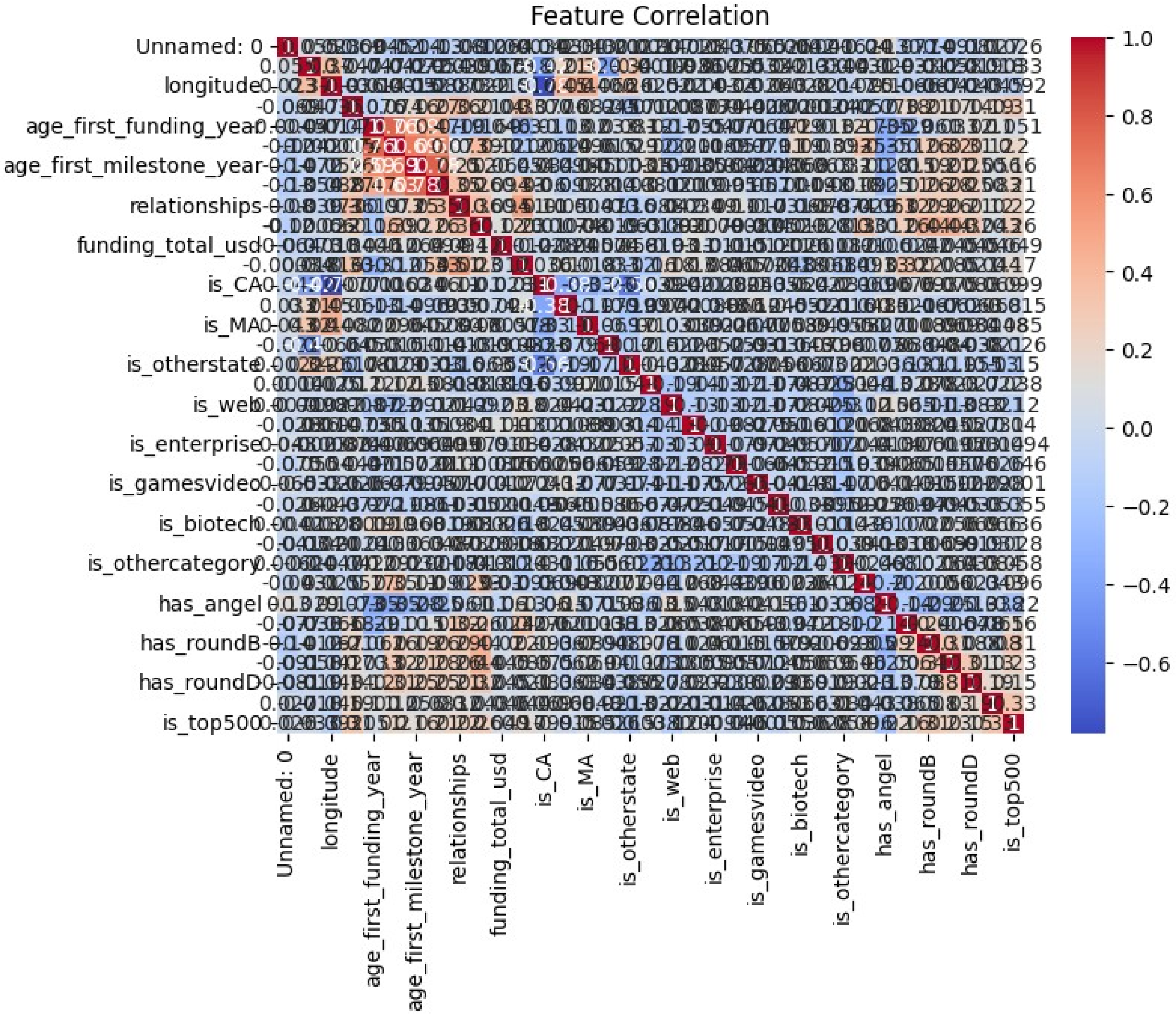
```
<IPython.core.display.HTML object>
```

```
Saving startup_success_prediction.xlsx to
startup_success_prediction.xlsx
Dataset Shape: (923, 49)

Missing Values:
 Unnamed: 0                    0
state_code                    0
latitude                      0
longitude                     0
zip_code                      0
id                            0
city                          0
Unnamed: 6                  493
name                          0
labels                        0
founded_at                    0
closed_at                   588
first_funding_at              0
last_funding_at               0
age_first_funding_year        0
age_last_funding_year         0
age_first_milestone_year    152
age_last_milestone_year     152
```

```
relationships                 0
funding_rounds                0
funding_total_usd             0
milestones                    0
state_code.1                  1
is_CA                         0
is_NY                         0
is_MA                         0
is_TX                         0
is_otherstate                 0
category_code                 0
is_software                   0
is_web                        0
is_mobile                     0
is_enterprise                 0
is_advertising                0
is_gamesvideo                 0
is_ecommerce                  0
is_biotech                    0
is_consulting                 0
is_othercategory              0
object_id                     0
has_VC                        0
has_angel                     0
has_roundA                    0
has_roundB                    0
has_roundC                    0
has_roundD                    0
avg_participants              0
is_top500                     0
status                        0
dtype: int64

Statistical Summary:
         Unnamed: 0    latitude   longitude      labels  \
count   923.000000  923.000000  923.000000  923.000000
mean    572.297941   38.517442 -103.539212    0.646804
std     333.585431    3.741497   22.394167    0.478222
min       1.000000   25.752358 -122.756956    0.000000
25%     283.500000   37.388869 -122.198732    0.000000
50%     577.000000   37.779281 -118.374037    1.000000
75%     866.500000   40.730646  -77.214731    1.000000
max    1153.000000   59.335232   18.057121    1.000000

       age_first_funding_year  age_last_funding_year  \
count              923.000000             923.000000
mean                 2.235630               3.931456
std                  2.510449               2.967910
min                 -9.046600              -9.046600
```

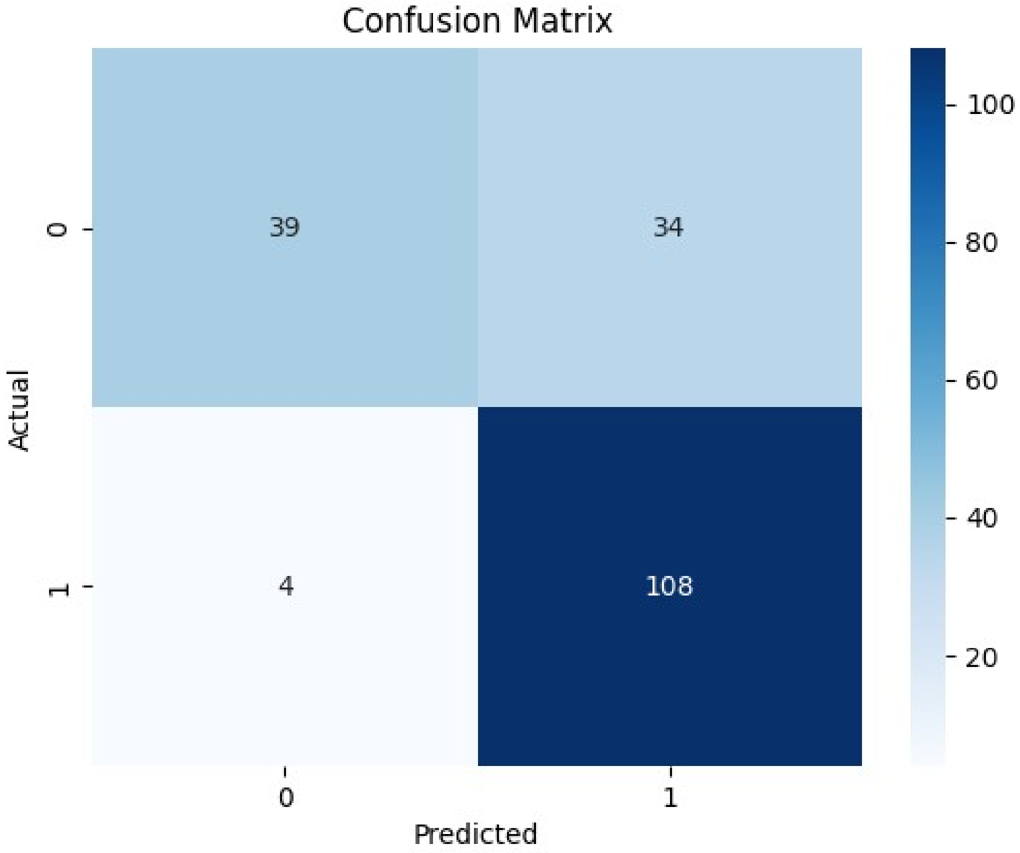
```
25%                  0.576700               1.669850
50%                  1.446600               3.528800
75%                  3.575350               5.560250
max                 21.895900              21.895900

       age_first_milestone_year  age_last_milestone_year
relationships  \
count                 771.000000               771.000000
923.000000
mean                    3.055353                 4.754423
7.710726
std                     2.977057                 3.212107
7.265776
min                   -14.169900                -7.005500
0.000000
25%                     1.000000                 2.411000
3.000000
50%                     2.520500                 4.476700
5.000000
75%                     4.686300                 6.753400
10.000000
max                    24.684900                24.684900
63.000000

       funding_rounds  ...  is_consulting  is_othercategory
has_VC  \
count      923.000000  ...     923.000000        923.000000
923.000000
mean         2.310943  ...       0.003250          0.322860
0.326111
std          1.390922  ...       0.056949          0.467823
0.469042
min          1.000000  ...       0.000000          0.000000
0.000000
25%          1.000000  ...       0.000000          0.000000
0.000000
50%          2.000000  ...       0.000000          0.000000
0.000000
75%          3.000000  ...       0.000000          1.000000
1.000000
max         10.000000  ...       1.000000          1.000000
1.000000

        has_angel  has_roundA  has_roundB  has_roundC  has_roundD  \
count  923.000000  923.000000  923.000000  923.000000  923.000000
mean     0.254605    0.508126    0.392199    0.232936    0.099675
std      0.435875    0.500205    0.488505    0.422931    0.299729
min      0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    1.000000    0.000000    0.000000    0.000000
```

```
75%      1.000000    1.000000    1.000000    0.000000    0.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000

      avg_participants   is_top500
count       923.000000  923.000000
mean          2.838586    0.809317
std           1.874601    0.393052
min           1.000000    0.000000
25%           1.500000    1.000000
50%           2.500000    1.000000
75%           3.800000    1.000000
max          16.000000    1.000000

[8 rows x 35 columns]
```



```
Best Parameters Found:
```

```
{'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 5, 'n_estimators': 200}

Training Accuracy: 0.9905149051490515
Testing Accuracy: 0.7945945945945946

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.53      0.67        73
           1       0.76      0.96      0.85       112

    accuracy                           0.79       185
   macro avg       0.83      0.75      0.76       185
weighted avg       0.82      0.79      0.78       185
```



```
Model & Scaler Saved Successfully!

<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>


Flask app.py file generated!
```