Smart Internz

Prosperity Prognosticator: Machine Learning for Startup Success Prediction

Category: Artificial Intelligence

1. Abstract

Startups play a vital role in economic growth and innovation, yet a large percentage fail due to poor planning, market misjudgement, or insufficient funding. **Prosperity Prognosticator** is a machine learning–based system designed to predict the likelihood of a startup's success by analysing key factors such as funding details, market trends, team composition, and business domain.
This project aims to assist **investors, entrepreneurs, and policymakers** in making data-driven decisions by forecasting startup success using predictive analytics.

2. Introduction

In today's competitive startup ecosystem, identifying potentially successful ventures at an early stage is challenging. Traditional decision-making methods rely heavily on intuition and experience, which may introduce bias and risk.

With the advancement of **machine learning and data analytics**, it is now possible to analyze historical startup data and extract meaningful patterns that influence success.
The **Prosperity Prognosticator** leverages machine learning algorithms to provide a reliable prediction model that evaluates startup viability.

3. Problem Statement

Many startups fail due to:

- Inadequate funding strategies
- Poor market fit
- Weak founding teams
- Lack of data-driven decision-making

There is a need for an **automated prediction system** that evaluates startup characteristics and predicts the probability of success, thereby reducing financial risks and improving strategic planning.

4. Objectives

The main objectives of this project are:

- To analyse historical startup datasets
- To identify key factors influencing startup success
- To build a machine learning model that predicts success probability
- To provide insights that help investors and entrepreneurs make informed decisions

5. Scope of the Project

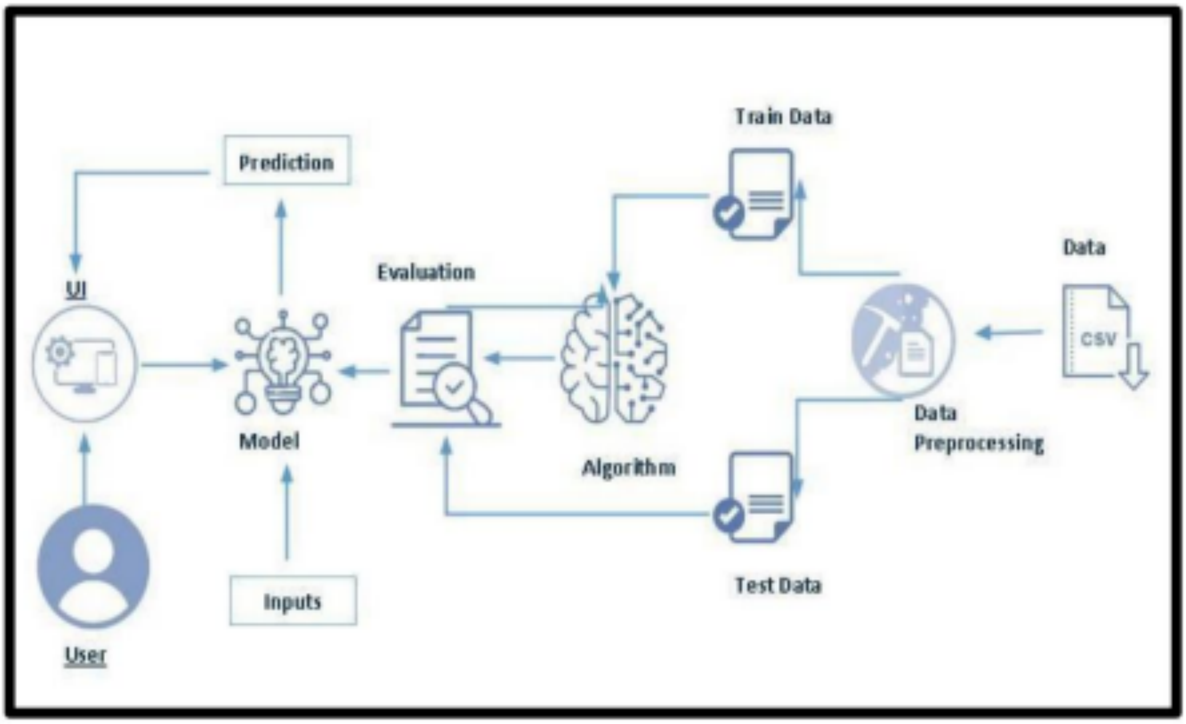
- Predict startup success or failure using machine learning
- Applicable to early-stage and growth-stage startups
- Useful for investors, incubators, and startup founders
- Can be extended to real-time market data in the future

6. System Architecture

Architecture Overview

1. Data Collection
2. Data Preprocessing
3. Feature Selection
4. Model Training
5. Prediction & Evaluation
6. Result Visualization

Technical Architecture:



7. Dataset Description

The dataset contains historical information about startups, including:

- Startup Name
- Industry / Domain
- Funding Amount
- Funding Rounds
- Market Category
- Team Size
- Year Founded
- Current Status (Successful / Failed)
- Link:https://www.kaggle.com/datasets/manishkc06/startup-success-prediction

8. Technologies Used

Programming Language

- Python

Libraries & Tools

- NumPy

- Pandas
- Scikit-learn
- Matplotlib
- Seaborn
- Jupyter Notebook

Machine Learning Algorithms

- Logistic Regression
- Decision Tree
- Random Forest

9. Methodology

9.1 Data Preprocessing

- Handling missing values
- Encoding categorical data
- Normalization and scaling

9.2 Feature Engineering

- Selection of influential features
- Removal of irrelevant attributes

9.3 Model Training

- Dataset split into training and testing sets
- Model trained using supervised learning

9.4 Model Evaluation

- Accuracy
- Precision
- Recall
- F1-score

10. Use Case Scenarios

Scenario 1: Investors

- Evaluate potential startup investments
- Reduce financial risk
- Optimize investment portfolio

Scenario 2: Entrepreneurs

- Assess readiness of their startup
- Identify weaknesses before launch
- Improve business strategies

Scenario 3: Policymakers

- Support innovation-friendly policies
- Identify high-potential startup sectors

11. Results and Discussion

The trained machine learning models successfully predict startup success with significant accuracy. Random Forest provided better performance compared to other algorithms due to its ability to handle non-linear relationships and feature importance analysis.

12. Advantages

- Data-driven decision-making
- Reduces investment risks
- Scalable and adaptable
- Easy to enhance with new data

13. Limitations

- Accuracy depends on dataset quality
- External economic factors not included
- Real-time market changes not considered

14. Future Enhancements

- Integration with real-time market data
- Deployment as a web-based application
- Use of deep learning models
- Addition of NLP-based analysis on startup descriptions

15. Conclusion

The **Prosperity Prognosticator** demonstrates how machine learning can be effectively used to predict startup success. By leveraging historical data and predictive analytics, the system provides valuable insights for investors, entrepreneurs, and decision-makers. This project highlights the importance of AI-driven solutions in modern business intelligence.

16. References

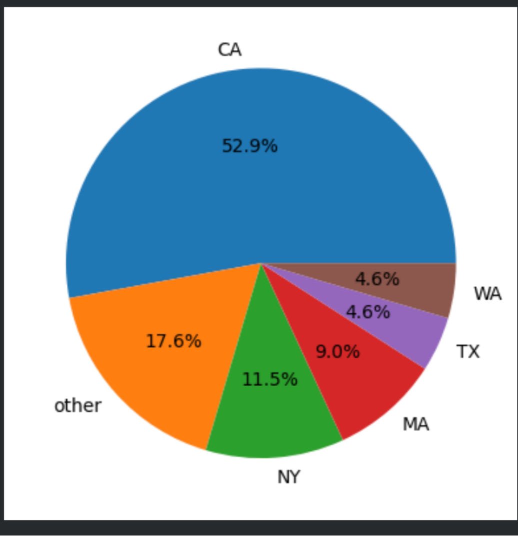
1. Scikit-learn Documentation
2. Kaggle Startup Datasets
3. Machine Learning by Tom Mitchell
4. Research Papers on Startup Analytics

OUTPUTS AND VISUALIZATIONS

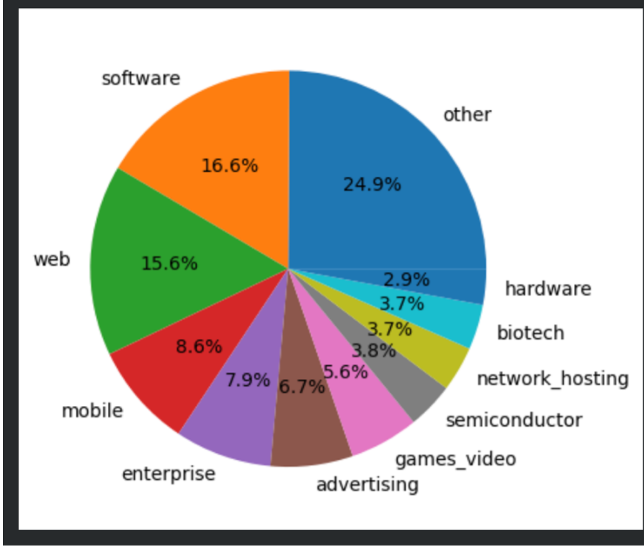
Univariate & Multivariate Analysis

```
data['State'] = 'other'
data.loc[(data['state_code'] == 'CA'), 'State'] = 'CA'
data.loc[(data['state_code'] == 'NY'), 'State'] = 'NY'
data.loc[(data['state_code'] == 'MA'), 'State'] = 'MA'
data.loc[(data['state_code'] == 'TX'), 'State'] = 'TX'
data.loc[(data['state_code'] == 'WA'), 'State'] = 'WA'
```

```
state_count = data['State'].value_counts()
plt.pie(state_count, labels = state_count.index, autopct = '%1.1f%%')
plt.show()
```




```
category_count = data['category'].value_counts()
plt.pie(category_count, labels = category_count.index, autopct = '%1.1f%%')
plt.show()
```

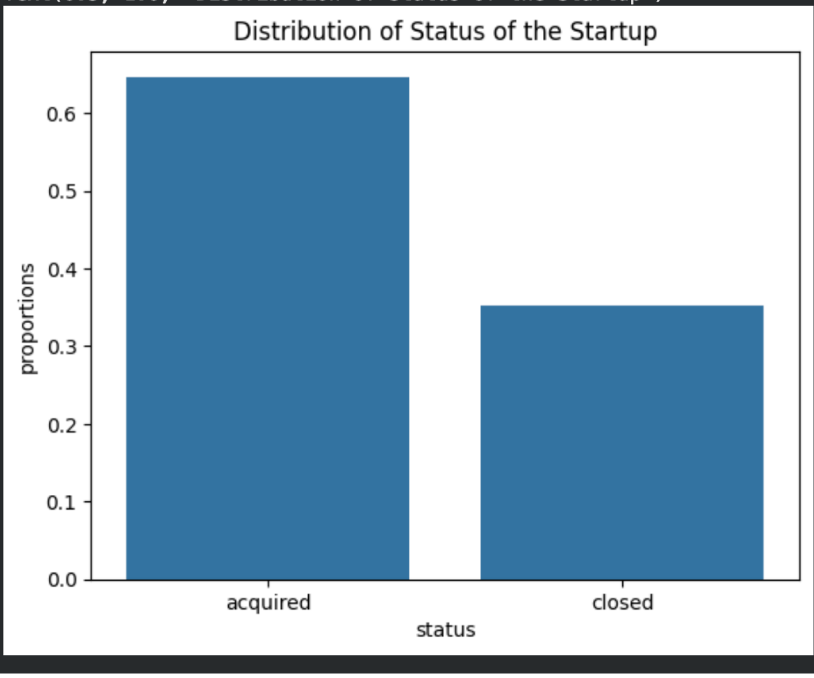


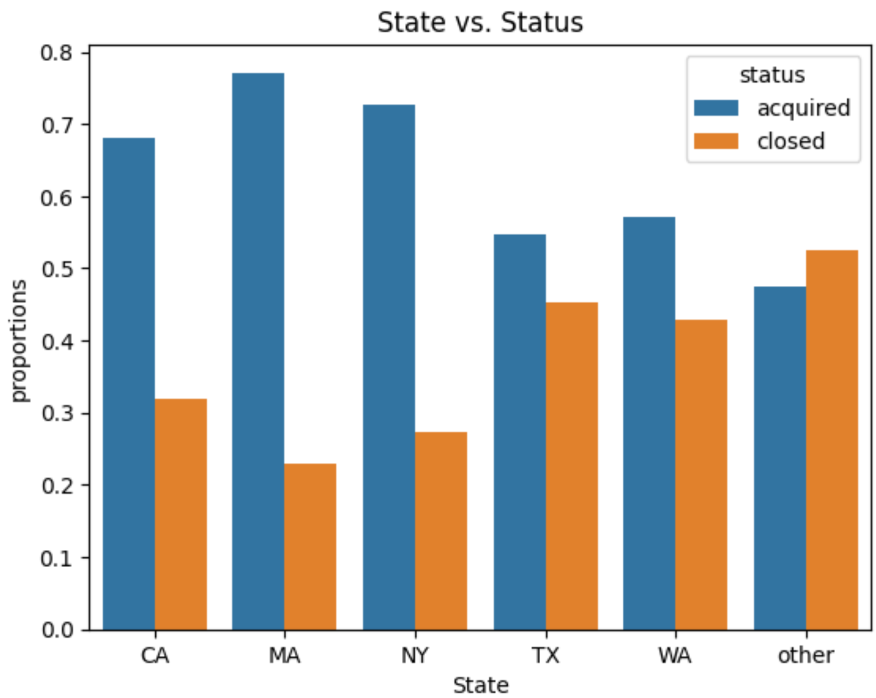
Distribution of Status of Startup

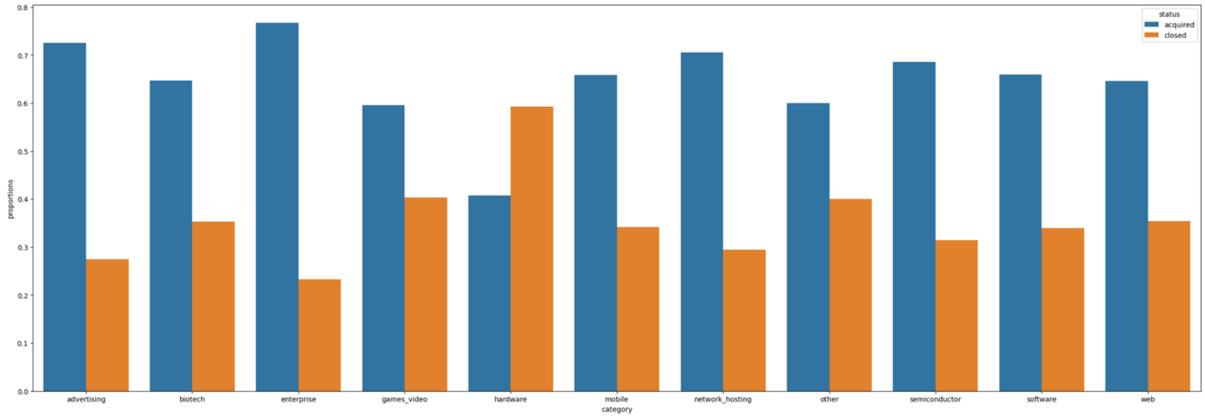
```
prop_df = data.groupby('status').size().reset_index(name = 'counts')
prop_df['proportions'] = prop_df['counts']/prop_df['counts'].sum()
```

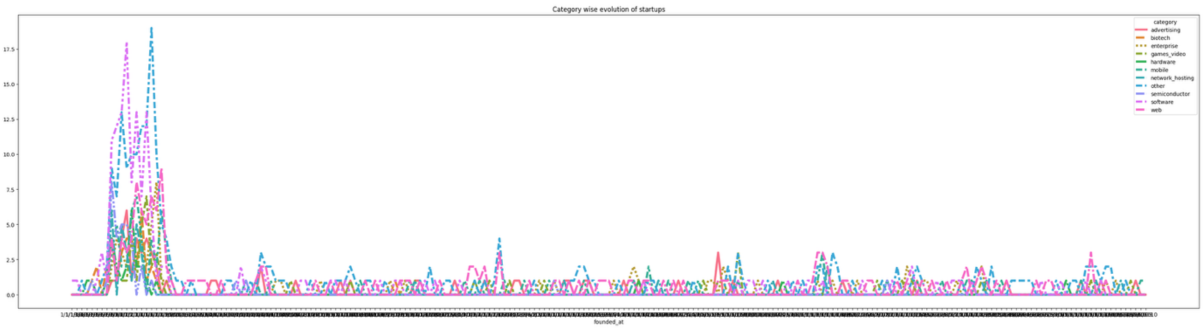
```
sns.barplot(data = prop_df, x = 'status', y = 'proportions')
plt.title('Distribution of Status of the Startup')
```

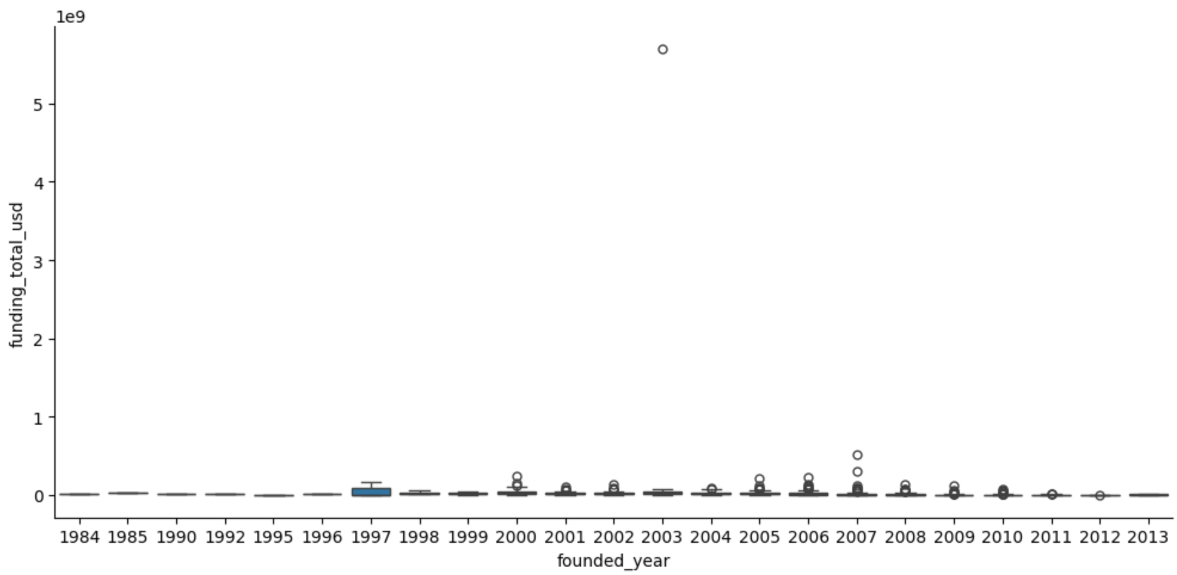
```
Text(0.5, 1.0, 'Distribution of Status of the Startup')
```






status
acquired
closed
proportions
advertising
biotech
enterprise
games_video
hardware
mobile
network_hosting
other
semiconductor
software
web
category

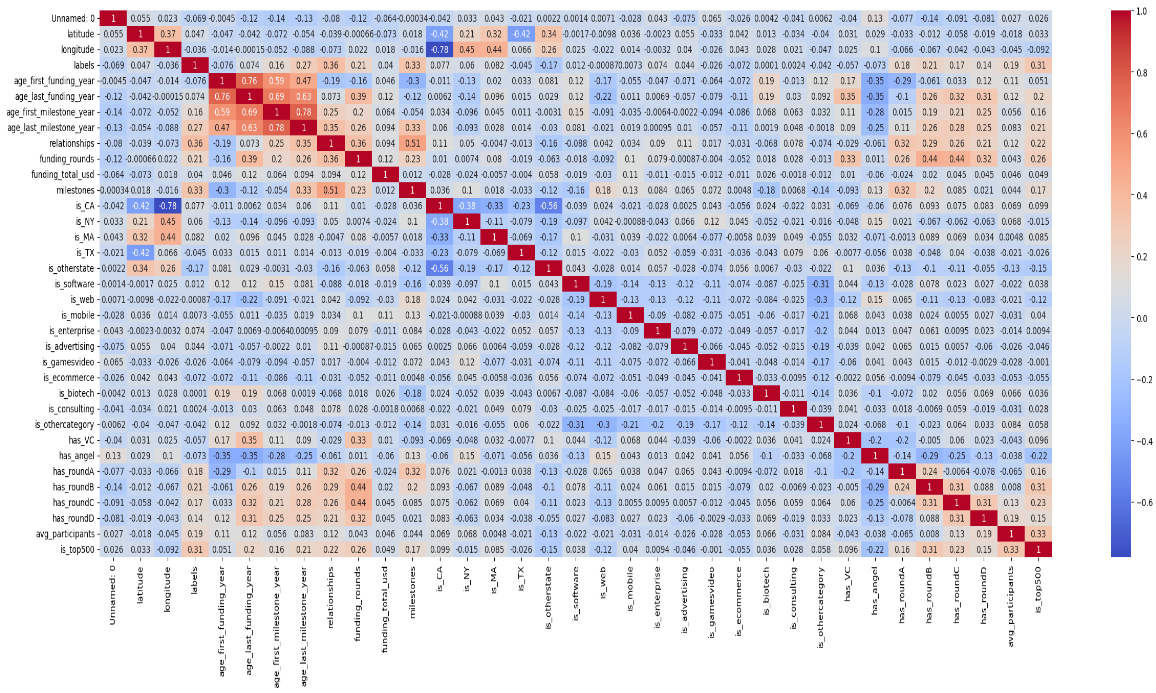

Category wise evolution of startups
category
advertising
biotech
enterprise
games_video
hardware
mobile
network_hosting
other
semiconductor
software
web
founded_at


1e9
funding_total_usd
1984 1985 1990 1992 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
founded_year

Statistical Analysis

```
data.describe(include=['float64','int64'])
```

	Unnamed: 0	latitude	longitude	labels	age_first_funding_year	age_last_funding_year	age_first_milestone_year	age_last_milestone_year	relationships
count	923.000000	923.000000	923.000000	923.000000	923.000000	923.000000	771.000000	771.000000	923.000000
mean	572.297941	38.517442	-103.539212	0.646804	2.235630	3.931456	3.055353	4.754423	7.710726
std	333.585431	3.741497	22.394167	0.478222	2.510449	2.967910	2.977057	3.212107	7.265776
min	1.000000	25.752358	-122.756956	0.000000	-9.046600	-9.046600	-14.169900	-7.005500	0.000000
25%	283.500000	37.388869	-122.198732	0.000000	0.576700	1.669850	1.000000	2.411000	3.000000
50%	577.000000	37.779281	-118.374037	1.000000	1.446600	3.528800	2.520500	4.476700	5.000000
75%	866.500000	40.730646	-77.214731	1.000000	3.575350	5.560250	4.686300	6.753400	10.000000
max	1153.000000	59.335232	18.057121	1.000000	21.895900	21.895900	24.684900	24.684900	63.000000



Reducing the Number of Categories

```
print(data['state_code'].equals(data['state_code.1']))
```

```
False
```

```
df =data.loc[data['state_code'] != data['state_code.1']]
df.style.set_properties(**{'background-color': 'yellow'}, subset=['state_code', 'state_code.1'])
```

	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed: 6	name	labels	founded_at	closed_at	first_funding_at	last_funding_at
515	1110	CA	37.451124	-122.166264	94025	c:856	Menlo Park	nan	Cuil	0	1/1/2005	9/1/2010	3/1/2007	4/15/2008

```python
state = data['state_code'].value_counts().to_frame()
state['proportion'] = state['count']/sum(state['count'])*100
state
```

state_code	count	proportion
CA	488	52.871073
NY	106	11.484290
MA	83	8.992416
WA	42	4.550379
TX	42	4.550379
CO	19	2.058505
IL	18	1.950163
PA	17	1.841820
VA	13	1.408451
GA	11	1.191766
NC	7	0.758397
OR	7	0.758397
NJ	7	0.758397
MD	7	0.758397
FL	6	0.650054
OH	6	0.650054

```python
rf = RandomForestClassifier(n_estimators=100, bootstrap=False, max_depth=20, min_samples_leaf=2, min_samples_split=2)
model_rf = rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)
cr_rf = classification_report(y_pred_rf, y_test)
print(cr_rf)
```

```
              precision    recall  f1-score   support

           0       0.48      0.80      0.60        54
           1       0.94      0.79      0.86       223

    accuracy                           0.79       277
   macro avg       0.71      0.79      0.73       277
weighted avg       0.85      0.79      0.81       277
```

```python
#checking accuracy
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
test_acc = accuracy_score(y_test,y_pred_test)
train_acc = accuracy_score(y_train,y_pred_train)
print('test_acc: ', test_acc)
print('train_acc:', train_acc)
```

```
test_acc:  0.7906137184115524
train_acc: 1.0
```

Save and load the best model

```python
import joblib
joblib.dump(model, 'random_forest_model.pkl')
```

```
['random_forest_model.pkl']
```