Artificial Intelligence

Prosperity Prognosticator: ML For Startup Success Prediction

1. AI/ML Overview

This project leverages Supervised Machine Learning to solve a binary classification problem: predicting whether a startup will succeed or fail. The system learns patterns from historical startup data and uses them to make predictions on new, unseen inputs.

2. ML Algorithms Used

Algorithm	Type	How It Works	Use in Project
Logistic Regression	**Linear Classifier**	**Models probability using sigmoid function**	**Baseline comparison**
Decision Tree	**Tree-based**	**Splits data on feature thresholds**	**Interpretable model**
Random Forest	**Ensemble**	**Aggregates multiple decision trees**	**Final selected model**
Gradient Boosting	**Ensemble**	**Sequentially improves weak learners**	**Performance benchmark**
SVM	**Kernel-based**	**Finds optimal hyperplane separating classes**	**High-dim comparison**
KNN	**Distance-based**	**Classifies based on k nearest neighbors**	**Simple baseline**

3. Model Training Process

3.1 Data Preparation

- Load startup_data.csv using pandas
- Drop irrelevant columns (id, name, zip_code)
- Fill missing values with column median
- Encode categorical columns using LabelEncoder
- Split data: 80% train, 20% test

3.2 Training and Evaluation

- Each model is trained using model.fit(X_train, y_train)
- Predictions made with model.predict(X_test)

- Metrics: Accuracy Score, Classification Report, Confusion Matrix
- Results compared in a bar chart

3.3 Hyperparameter Tuning

- GridSearchCV applied to Random Forest
- Parameters tuned: n_estimators, max_depth, min_samples_split
- 5-fold cross-validation used for reliable evaluation
- Best parameters automatically selected

3.4 Feature Importance

- Random Forest provides feature_importances_ scores
- Features ranked and top 10 selected
- Final model retrained with top 10 features
- Accuracy before and after compared

4. Model Performance Metrics

Metric	Description	Formula
Accuracy	**Percentage of correct predictions**	**(TP + TN) / Total**
Precision	**Of predicted positives, how many are correct**	**TP / (TP + FP)**
Recall	**Of actual positives, how many were caught**	**TP / (TP + FN)**
F1 Score	**Harmonic mean of precision and recall**	**2 * (P * R) / (P + R)**

5. Model Saving and Loading

After training and tuning, the best model is saved using Python's Pickle library:

- Saving: pickle.dump(model, open('random_forest_model.pkl', 'wb'))
- Loading in Flask: pickle.load(open('random_forest_model.pkl', 'rb'))
- Feature list saved: pickle.dump(features, open('features.pkl', 'wb'))