Titanic Survival Prediction Model

■ Overview

This project builds a machine learning model to predict whether a passenger survived the Titanic disaster using passenger data (age, sex, class, fare, etc.). Implemented in a Jupyter Notebook, the workflow covers data exploration, preprocessing, feature engineering, model training, evaluation, and interpretation.

■ Dataset

The project uses the classic Kaggle 'Titanic — Machine Learning from Disaster' dataset. The dataset includes fields such as PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked.

■ Workflow

1. Load data into pandas DataFrame.
2. Exploratory Data Analysis (EDA): check missing values, distributions, and correlations.
3. Data Cleaning: impute missing ages, drop or handle Cabin, fill Embarked, fix data types.
4. Feature Engineering: extract Title from Name, create FamilySize, IsAlone, AgeBins, FareBins, encode categorical features.
5. Feature Selection: choose relevant features such as Pclass, Sex, Age, Fare, Embarked, Title, FamilySize.
6. Model Training: split data into train/test sets and train models (Logistic Regression, Random Forest, XGBoost).
7. Hyperparameter Tuning: use GridSearchCV or RandomizedSearchCV.
8. Evaluation: report accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix.
9. Interpretation: SHAP values or feature importances to explain model predictions.
10. Save model and notebook for reproducibility.

■ Libraries & Tools

Python, Jupyter Notebook, pandas, numpy, scikit-learn, matplotlib, seaborn (optional for visuals), XGBoost (optional), SHAP (optional), joblib/pickle for saving models.

■■ Preprocessing Steps (detailed)

- Handle missing Age using median or predictive imputation.
- Fill missing Embarked with mode.
- Create Title feature from Name (Mr, Mrs, Miss, Master, Rare).
- Convert Sex and Embarked to numeric using One-Hot or Label Encoding.
- Create FamilySize = SibSp + Parch + 1 and IsAlone feature.
- Bin Age and Fare if needed to reduce skew.
- Scale numerical features with StandardScaler or MinMaxScaler for models that require it.

■ Model Options & Why

- Logistic Regression: strong baseline, interpretable coefficients.
- Random Forest: handles nonlinearities and interactions, robust to outliers.
- XGBoost: often delivers top performance with tabular data.
- SVM/KNN: alternatives for experimentation.

■ Evaluation Metrics

Accuracy, Precision, Recall, F1-score, ROC-AUC, and Confusion Matrix. For imbalanced settings, prioritize F1-score and ROC-AUC.

■ Example Code Snippet (feature engineering & training)

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# load
df = pd.read_csv('train.csv')
# basic feature engineering
(df['FamilySize'] = df['SibSp'] + df['Parch'] + 1)
df['IsAlone'] = (df['FamilySize'] == 1).astype(int)
df['Title'] = df['Name'].str.extract(',\s*([^.]*)\.')
# select features
features = ['Pclass','Sex','Age','Fare','Embarked','FamilySize','IsAlone','Title']
# ... (encoding & imputation steps)
X = df[features]
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
print(classification_report(y_test, clf.predict(X_test)))
```

■■ Project Structure

```
/titanic_project
■■ data/
■ ■■ train.csv
■ ■■ test.csv
■■ notebooks/
■ ■■ titanic_exploration.ipynb
■■ models/
■ ■■ rf_model.joblib
■■ src/
■ ■■ preprocess.py
■■ requirements.txt
■■ README.md
```

```

## ■ Results & Findings

Report baseline and final model metrics. Common observations: females and higher class passengers had higher survival rates; family size and title are predictive.

## ■ How to Run

1. Create virtual environment and install requirements: `pip install -r requirements.txt`.
2. Place `train.csv` in `data/`.
3. Run the notebook or `python src/train.py` to train and evaluate.
4. Use saved model to make predictions on `test.csv`.

## ■ Possible Improvements

- Use cross-validation and more robust hyperparameter tuning.
- Try ensemble stacking of multiple models.
- Use SHAP or LIME for deeper explanation.
- Feature creation like ticket frequency or cabin deck extraction.

## ■ References

Kaggle Titanic competition dataset and discussion kernels; scikit-learn documentation.
```