3. Titanic Survival Prediction

Description: Predict whether a passenger survived the Titanic disaster based on features like age, class, gender, etc.

Steps: -

•Load the Titanic dataset (available on Kaggle or Seaborn).

•Perform data cleaning and feature engineering.

•Convert categorical variables using One-Hot Encoding.

•Train a Logistic Regression model.

•Evaluate using accuracy, precision, recall, and confusion matrix.

Project Code

titanic_survival_prediction/main.py

```python
# titanic_survival_prediction/main.py
from src.data_preprocessing import load_data, clean_data, encode_features
from src.model_training import train_and_evaluate
import os


if __name__ == "__main__":
    os.makedirs("models", exist_ok=True)

    df = load_data("data/titanic.csv")
    if df.empty:
        raise ValueError("The dataset is empty. Please check the file path and content.")

    # Clean data
    df = clean_data(df)

    # Detect correct target column
    target_col = 'survived' if 'survived' in df.columns else 'Survived'

    # Split features and target
    X = df.drop(target_col, axis=1)
    y = df[target_col]

    # Encode categorical features
    X, encoder = encode_features(X)

    # Train and evaluate model
    train_and_evaluate(X, y)
```

titanic_survival_prediction/src/model_training.py

```python
# titanic_survival_prediction/src/model_training.py
import joblib
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
from .utils import plot_confusion_matrix

def train_and_evaluate(X, y):
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    print(f"Accuracy: {acc:.2f}")
    print(f"Precision: {prec:.2f}")
    print(f"Recall: {rec:.2f}")
    print("Confusion Matrix:")
    print(cm)

    plot_confusion_matrix(cm)

    joblib.dump(model, "models/logistic_model.pkl")
    print("✅ Model saved in 'models/logistic_model.pkl'")
```

titanic_survival_prediction/src/utils.py

```python
# titanic_survival_prediction/src/utils.py
import seaborn as sns
import matplotlib.pyplot as plt

def plot_confusion_matrix(cm):
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title("Confusion Matrix")
    plt.show()
```

Project Code Output

```
  n_iter_i = _check_optimize_result(
Accuracy: 0.82
Precision: 0.81
Recall: 0.74
Confusion Matrix:
[[92 13]
 [19 55]]
✅ Model saved in 'models/logistic_model.pkl'
```