```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc

import matplotlib.pyplot as plt


# Load dataset

data = pd.read_csv('EX4B.csv')


# Convert target column to binary (1 for On Time, 0 for Late)

data['Journey Status'] = data['Journey Status'].apply(lambda x: 1 if x == 'On Time' else 0)


# Convert time columns to seconds

def time_to_seconds(t):

    h, m, s = map(int, t.split(':'))

    return h * 3600 + m * 60 + s


data['Departure Time'] = data['Departure Time'].apply(time_to_seconds)

data['Arrival Time'] = data['Arrival Time'].apply(time_to_seconds)


# Features and target

attributes = data[['Price', 'Ticket Class', 'Departure Time', 'Arrival Time']]

target = data['Journey Status']


# Preprocessing

preprocess = ColumnTransformer([

    ('num', StandardScaler(), ['Price', 'Departure Time', 'Arrival Time']),
    ('cat', OneHotEncoder(), ['Ticket Class'])
])

# Train-test split
attributes_train, attributes_test, target_train, target_test = train_test_split(
    attributes, target, test_size=0.2, random_state=42)

# Apply preprocessing and train model
attributes_train = preprocess.fit_transform(attributes_train)
attributes_test = preprocess.transform(attributes_test)

model = LogisticRegression()
model.fit(attributes_train, target_train)

# Predict
target_pred = model.predict(attributes_test)

# Evaluation
print("Accuracy:", accuracy_score(target_test, target_pred))
print("Confusion Matrix:\n", confusion_matrix(target_test, target_pred))
print("Classification Report:\n", classification_report(target_test, target_pred))

# ROC Curve
target_prob = model.predict_proba(attributes_test)[:, 1]  # Get probabilities for class 1
fpr, tpr, _ = roc_curve(target_test, target_prob)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend()

plt.show()
```