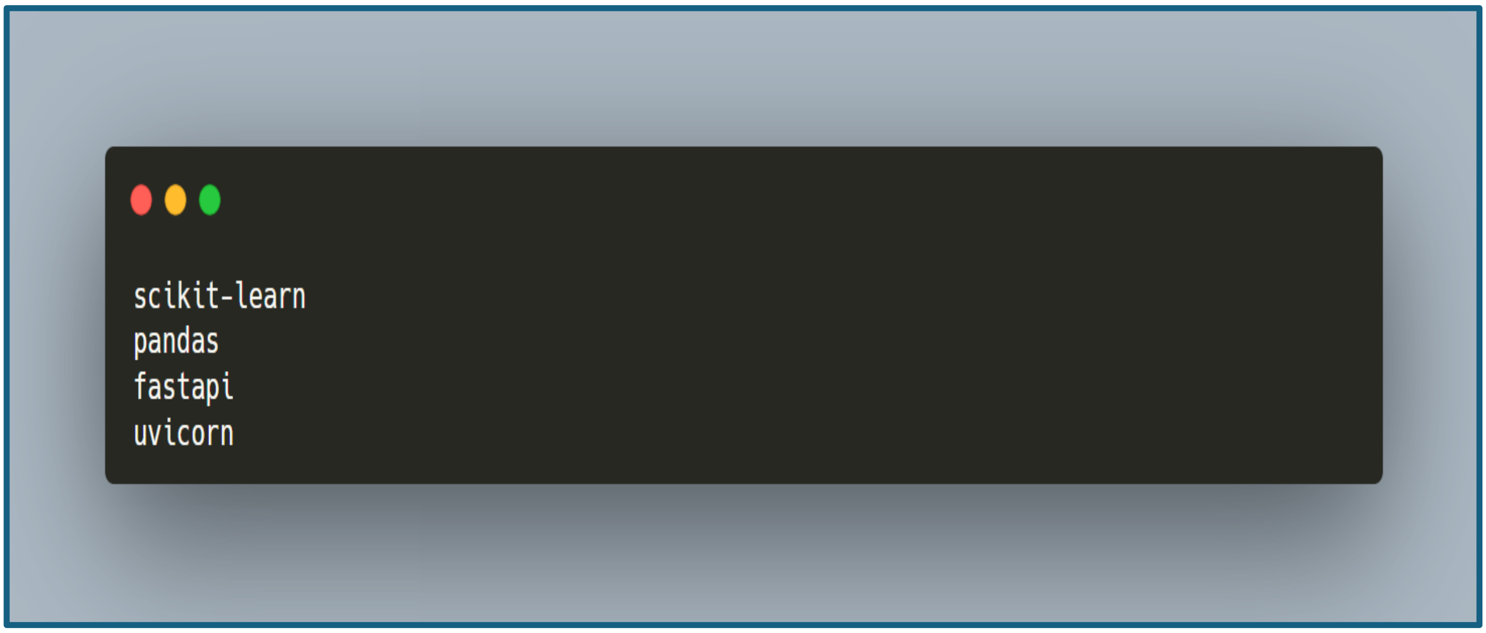
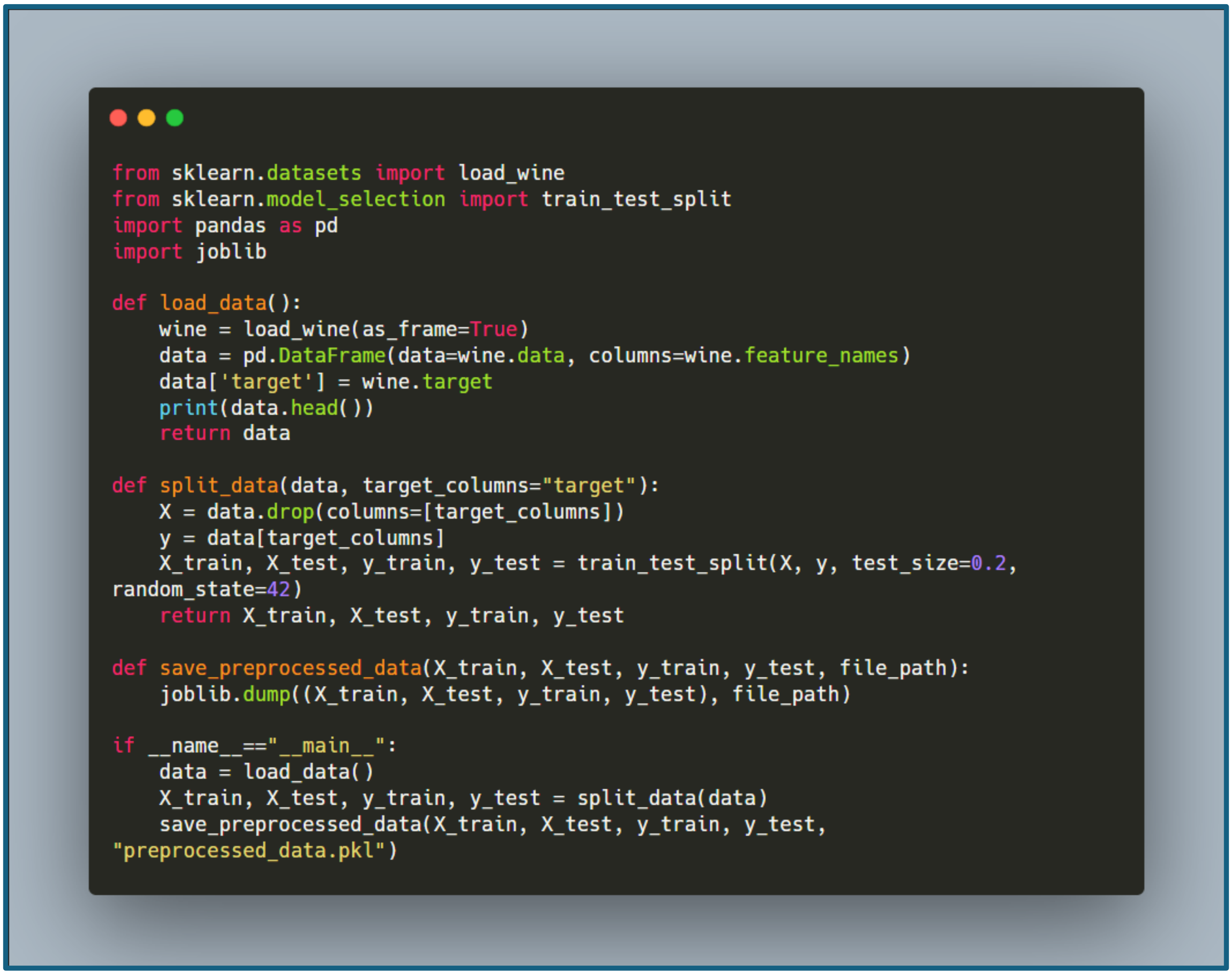


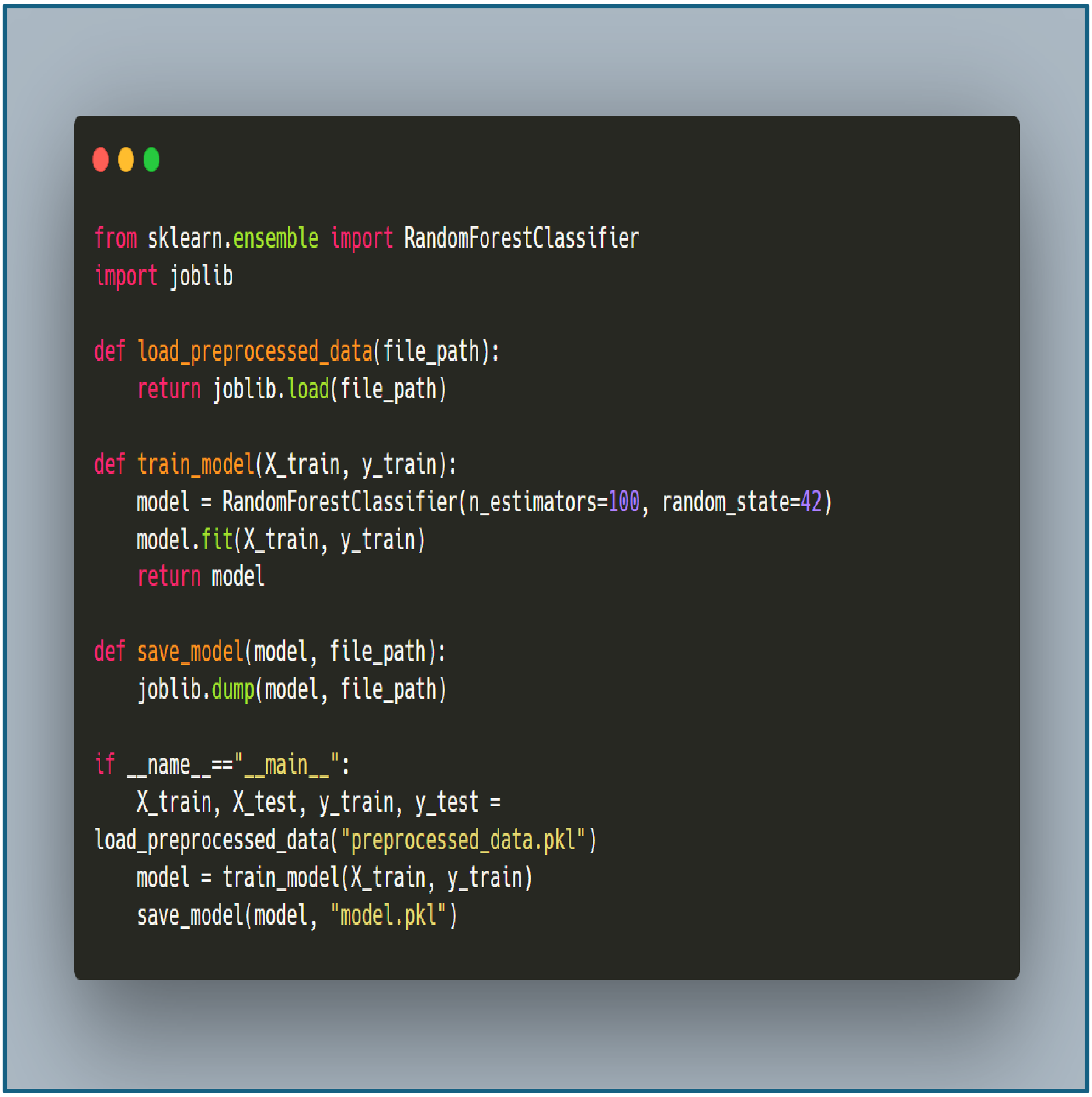
```
scikit-learn
pandas
fastapi
uvicorn
```

requirements.txt



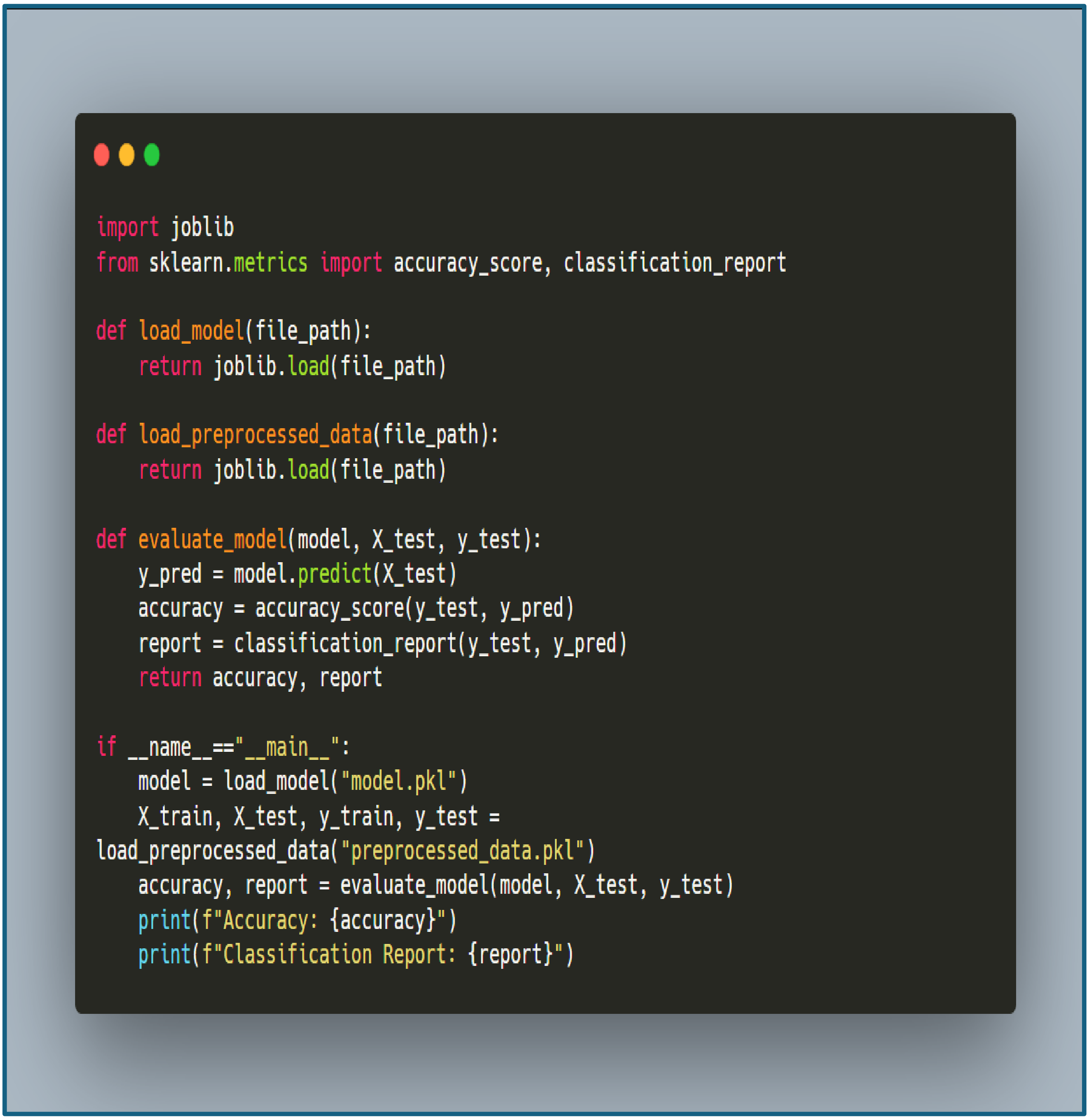
```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
import pandas as pd
import joblib

def load_data():
    wine = load_wine(as_frame=True)
    data = pd.DataFrame(data=wine.data, columns=wine.feature_names)
    data['target'] = wine.target
    print(data.head())
    return data

def split_data(data, target_columns="target"):
    X = data.drop(columns=[target_columns])
    y = data[target_columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    return X_train, X_test, y_train, y_test

def save_preprocessed_data(X_train, X_test, y_train, y_test, file_path):
    joblib.dump((X_train, X_test, y_train, y_test), file_path)

if __name__=="__main__":
    data = load_data()
    X_train, X_test, y_train, y_test = split_data(data)
    save_preprocessed_data(X_train, X_test, y_train, y_test,
"preprocessed_data.pkl")
```

data_loading.py



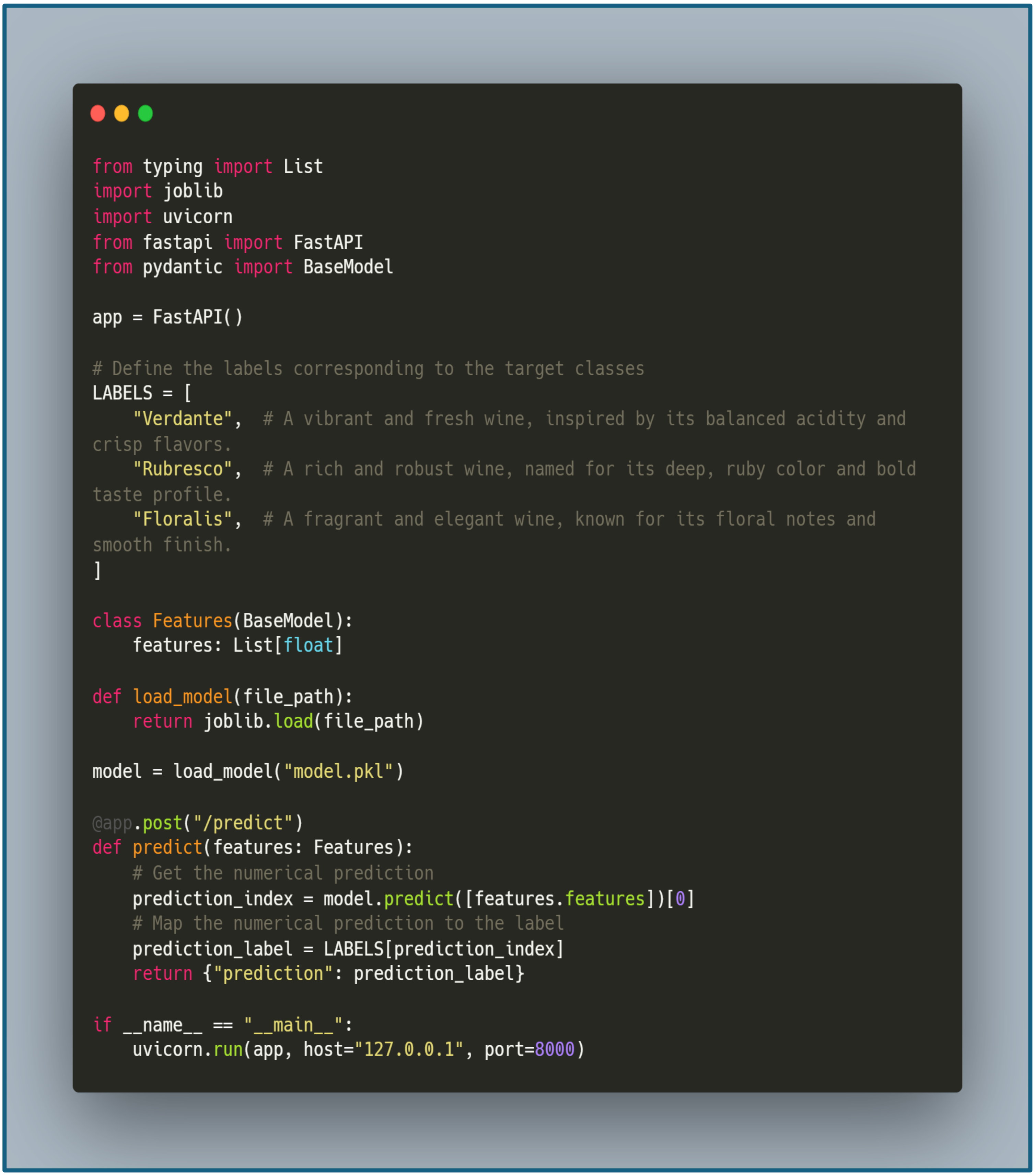
```python
from sklearn.ensemble import RandomForestClassifier
import joblib

def load_preprocessed_data(file_path):
    return joblib.load(file_path)

def train_model(X_train, y_train):
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    return model

def save_model(model, file_path):
    joblib.dump(model, file_path)

if __name__=="__main__":
    X_train, X_test, y_train, y_test =
load_preprocessed_data("preprocessed_data.pkl")
    model = train_model(X_train, y_train)
    save_model(model, "model.pkl")
```

model_training.py



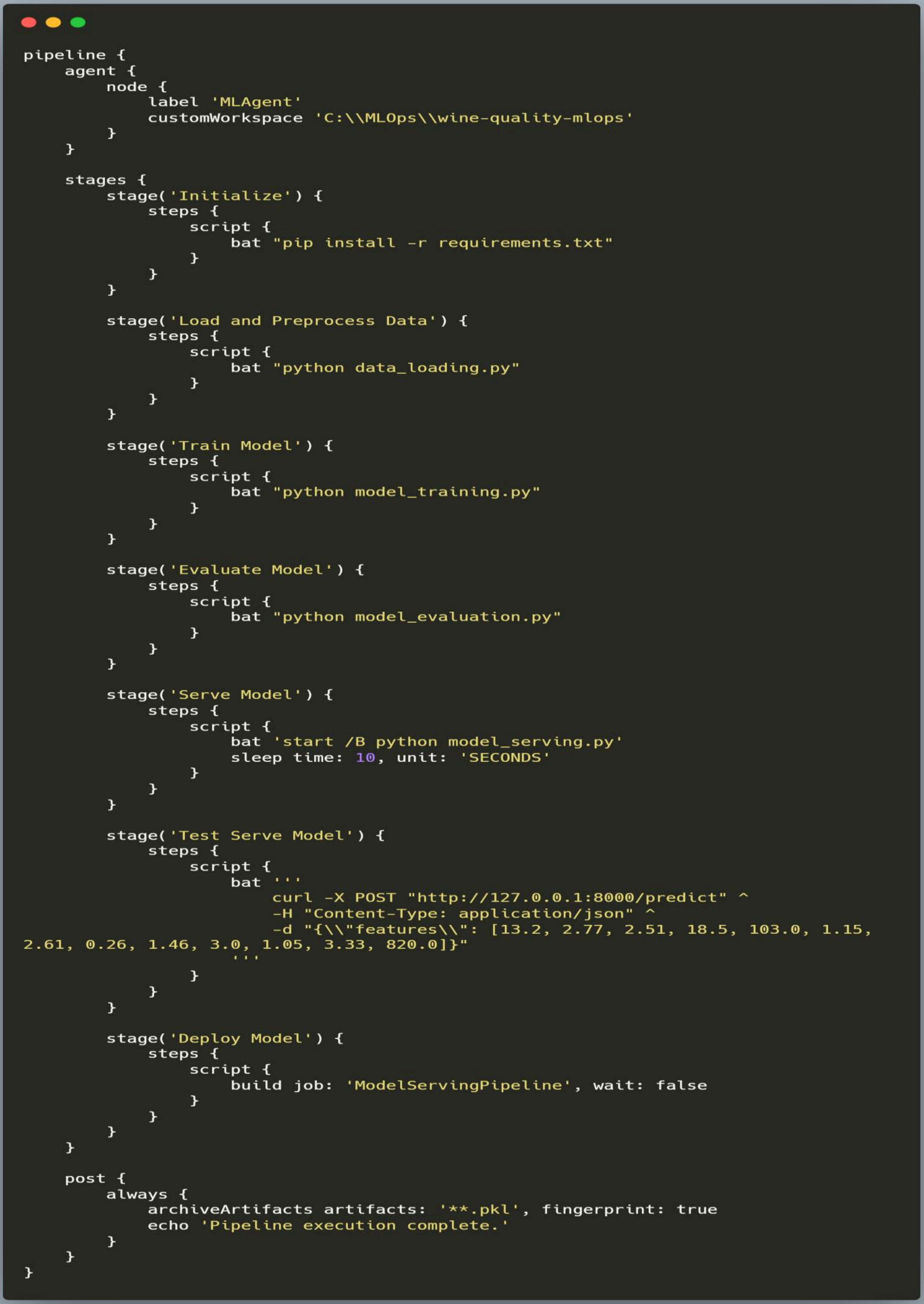
```python
import joblib
from sklearn.metrics import accuracy_score, classification_report

def load_model(file_path):
    return joblib.load(file_path)

def load_preprocessed_data(file_path):
    return joblib.load(file_path)

def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    return accuracy, report

if __name__=="__main__":
    model = load_model("model.pkl")
    X_train, X_test, y_train, y_test =
load_preprocessed_data("preprocessed_data.pkl")
    accuracy, report = evaluate_model(model, X_test, y_test)
    print(f"Accuracy: {accuracy}")
    print(f"Classification Report: {report}")
```

model_evaluation.py



```python
from typing import List
import joblib
import uvicorn
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

# Define the labels corresponding to the target classes
LABELS = [
    "Verdante",  # A vibrant and fresh wine, inspired by its balanced acidity and
crisp flavors.
    "Rubresco",  # A rich and robust wine, named for its deep, ruby color and bold
taste profile.
    "Floralis",  # A fragrant and elegant wine, known for its floral notes and
smooth finish.
]

class Features(BaseModel):
    features: List[float]

def load_model(file_path):
    return joblib.load(file_path)

model = load_model("model.pkl")

@app.post("/predict")
def predict(features: Features):
    # Get the numerical prediction
    prediction_index = model.predict([features.features])[0]
    # Map the numerical prediction to the label
    prediction_label = LABELS[prediction_index]
    return {"prediction": prediction_label}

if __name__ == "__main__":
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

model_serving.py



```
pipeline {
    agent {
        node {
            label 'MLAgent'
            customWorkspace 'C:\\MLOps\\wine-quality-mlops'
        }
    }

    stages {
        stage('Initialize') {
            steps {
                script {
                    bat "pip install -r requirements.txt"
                }
            }
        }

        stage('Load and Preprocess Data') {
            steps {
                script {
                    bat "python data_loading.py"
                }
            }
        }

        stage('Train Model') {
            steps {
                script {
                    bat "python model_training.py"
                }
            }
        }

        stage('Evaluate Model') {
            steps {
                script {
                    bat "python model_evaluation.py"
                }
            }
        }

        stage('Serve Model') {
            steps {
                script {
                    bat 'start /B python model_serving.py'
                    sleep time: 10, unit: 'SECONDS'
                }
            }
        }

        stage('Test Serve Model') {
            steps {
                script {
                    bat '''
                        curl -X POST "http://127.0.0.1:8000/predict" ^
                        -H "Content-Type: application/json" ^
                        -d "{\\"features\\": [13.2, 2.77, 2.51, 18.5, 103.0, 1.15,
2.61, 0.26, 1.46, 3.0, 1.05, 3.33, 820.0]}"
                    '''
                }
            }
        }

        stage('Deploy Model') {
            steps {
                script {
                    build job: 'ModelServingPipeline', wait: false
                }
            }
        }
    }

    post {
        always {
            archiveArtifacts artifacts: '**.pkl', fingerprint: true
            echo 'Pipeline execution complete.'
        }
    }
}
```

Jenkins file