

目录

python 程序设计大作业	1
计算机科学与工程学院	1
1. 概述	1
1.1 问题描述	1
1.2 功能说明	1
1.2.1 基本功能	1
1.2.2 附加功能（flask 框架 web 网页与可视化大屏）	7
1.3 技术路线	8
2. 相关技术	10
2.1 爬虫	10
2.2 Pandas 操作数据	10
2.3 Matplotlib 绘制图形	10
2.4 Sklearn 构建机器学习模型	11
2.5 Flask 框架构建 web 平台	11
3. 编码实现	12
3.1 爬虫爬取天气	12
3.2 pandas 读取数据、合并数据	13
3.3 matplotlib 绘制图形	14
3.4 sklearn 构建机器学习模型与 flask 构建 web 平台	14
4. 遇到的主要问题及解决方法	18
5. 总结与展望	18

1. 概述

1.1 问题描述

航空行业是全球经济的重要组成部分，航班票价的波动直接影响着旅客的出行计划和航空公司的盈利能力。然而，航班票价受到诸多因素的影响，包括但不限于航线、出发时间、舱位等级、节假日以及经济因素等。因此，如何准确预测航班票价，以及如何对影响票价的因素进行分析，对于航空公司和旅客来说都具有重要意义。

本次 Python 大作业，我基于这个主题，使用 Python 的 pandas、matplotlib、sklearn、requests、scipy、flask 等库，在 github 公开的数据集上以及爬取得到的天气数据整合为合并数据集，通过随机森林方法预测统一航线的不同班次票价情况，并构建一个可视化与预测查询平台，意在为航空出行提供方便，把握票价波动。

1.2 功能说明

1.2.1 基本功能

☆数据爬取: 通过 Python 的 requests 库及正则表达式技术爬取印度新德里 2023 全年的数据。如下图所示，即为需要爬取的网站界面：



图 1 需要爬取的网站界面

☆数据存储: 爬取的数据存放在 excel 电子表格中，与在 github 上获得的开源数据集通过 pandas 合并为总数据集，存放在 excel 电子表格中。开源数据集主要包括 'Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arrival_Time', 'Duration',

'Total_Stops', 'Additional_Info', 'Price'等字段，分别表示航空公司、日期、出发地、目的地、路线、出发时间、到达时间、持续时间、停站数、服务质量、价格，我们选取数据中的新德里到科钦的所有航班，然后将其与爬取的天气数据（天气类型、风向、最高温度、最低温度）通过 pandas 操作对应起来，得到总数据集。如下图，在 jupyter 中读取总数据集的展示：

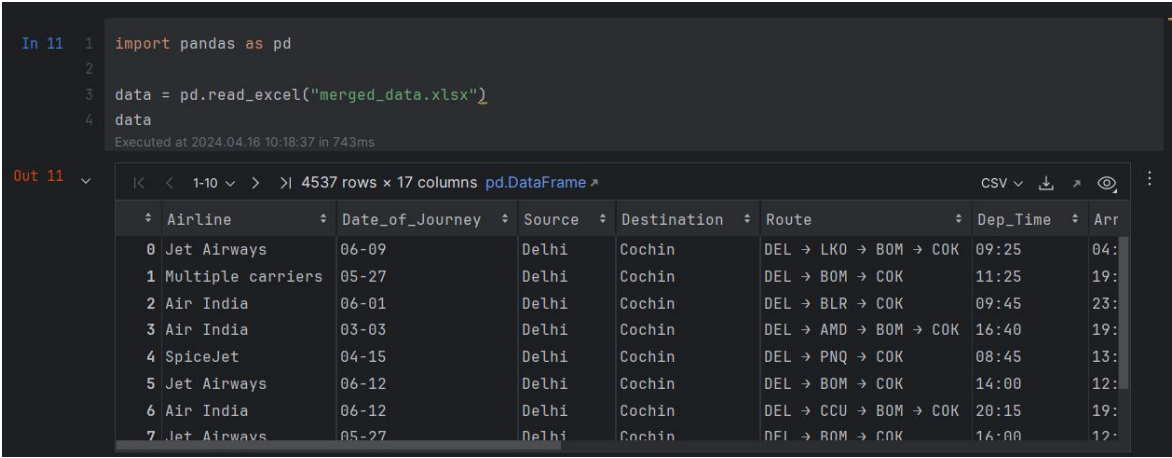


图 2 在 jupyter 中读取存储的 excel 数据

☆数据分析与可视化展示：调用 matplotlib 库中的 pyplot 对象在 jupyter 中绘制了航空公司与票价的关系、日期与均价的关系、路线与票价的关系、出发时间与票价关系、持续时间与票价的关系、航班服务与票价的关系及分布情况、最高温度与票价的关系、最低温度与票价的关系、风向与票价的关系、降水量与票价的关系、风向与票价的关系、天气与票价的关系与分布情况。

(1) 航空公司与票价

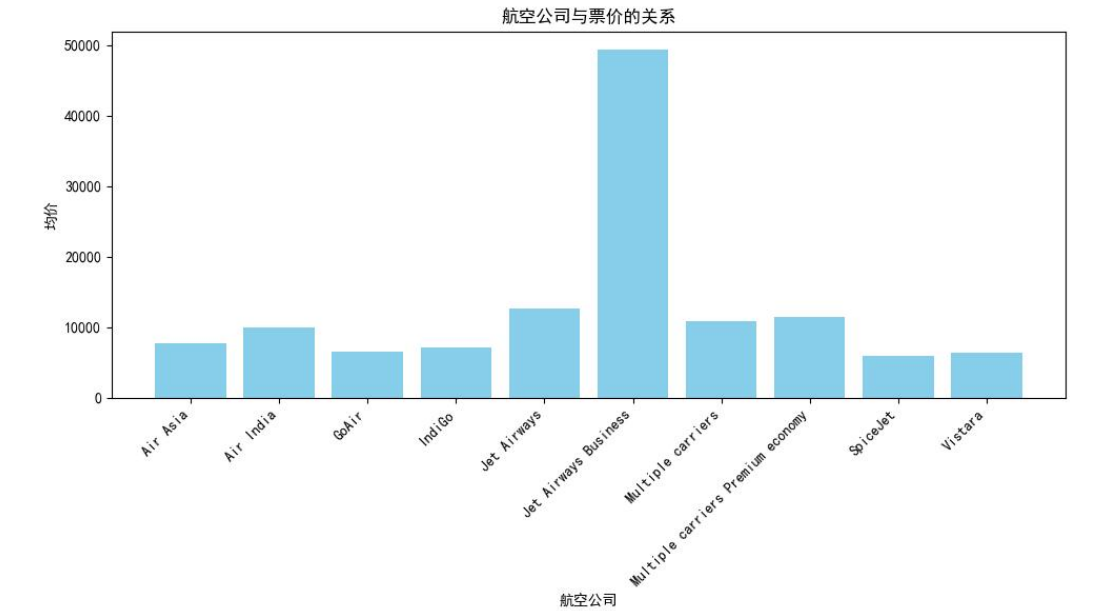


图 3 航空公司与票价

(2) 日期与均价

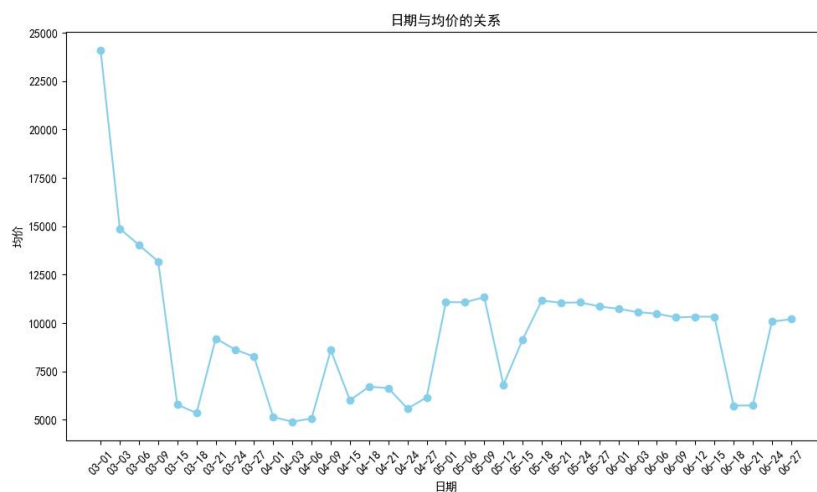


图 4 日期与均价

(3) 路线与票价

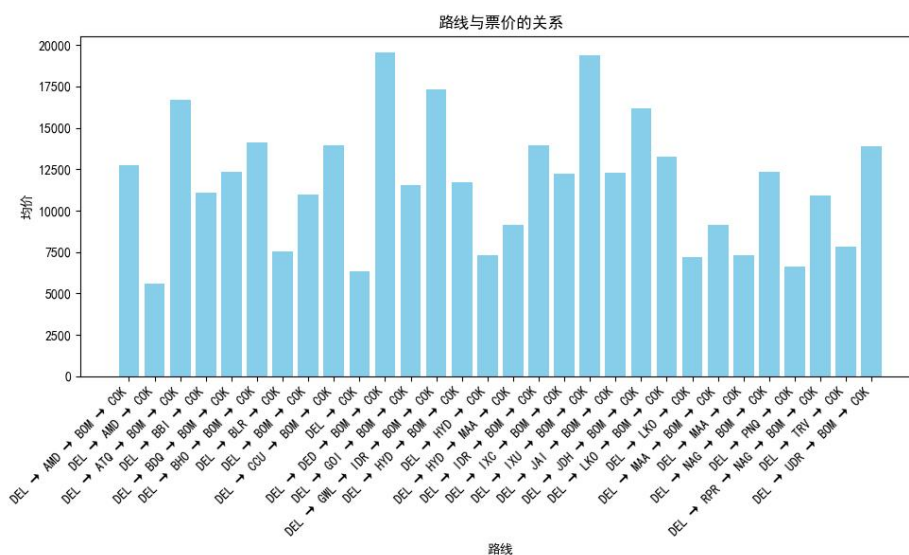


图 5 路线与票价

(4) 出发时间与票价

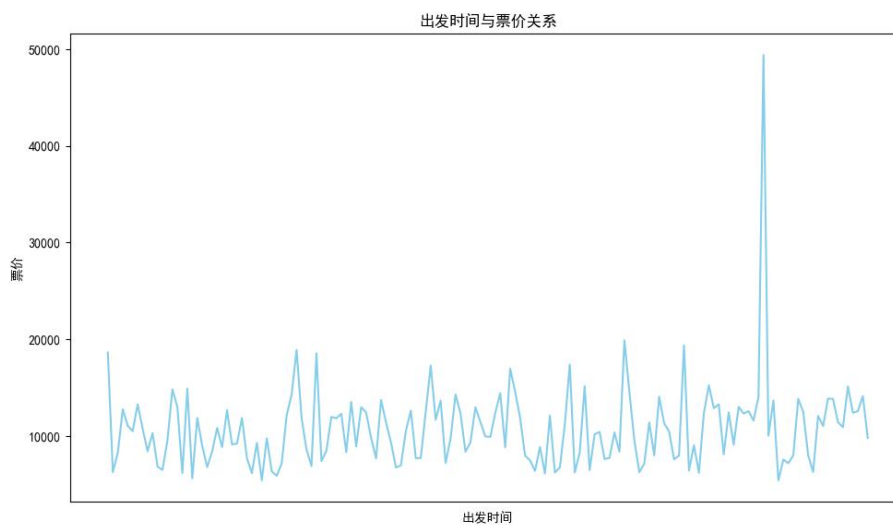
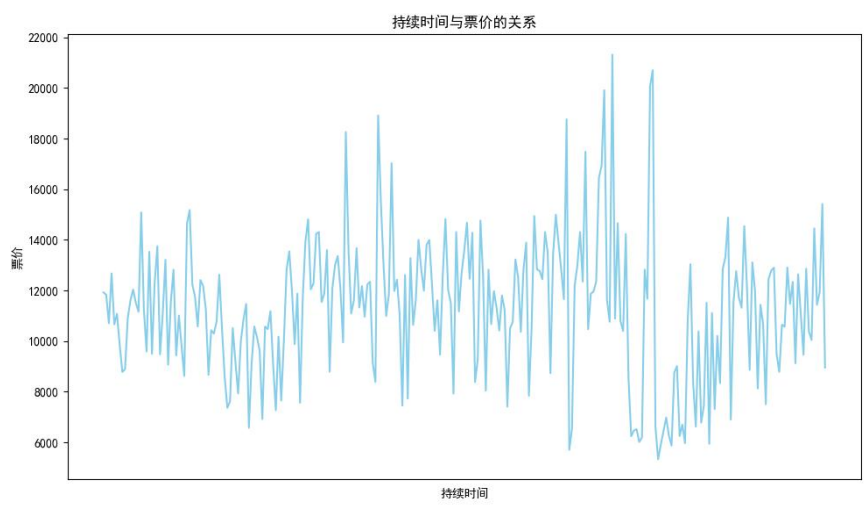
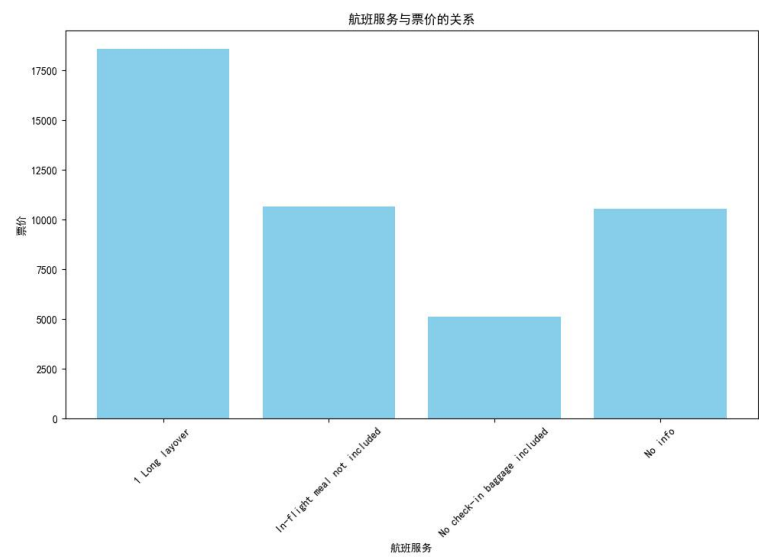
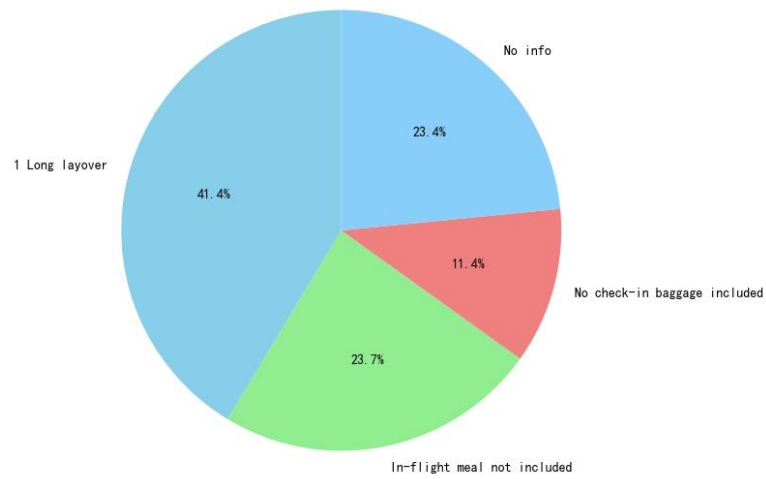


图 6 出发时间与票价

(5) 持续时间与票价



(6) 航班服务分布及与票价关系



(7) 温度与票价

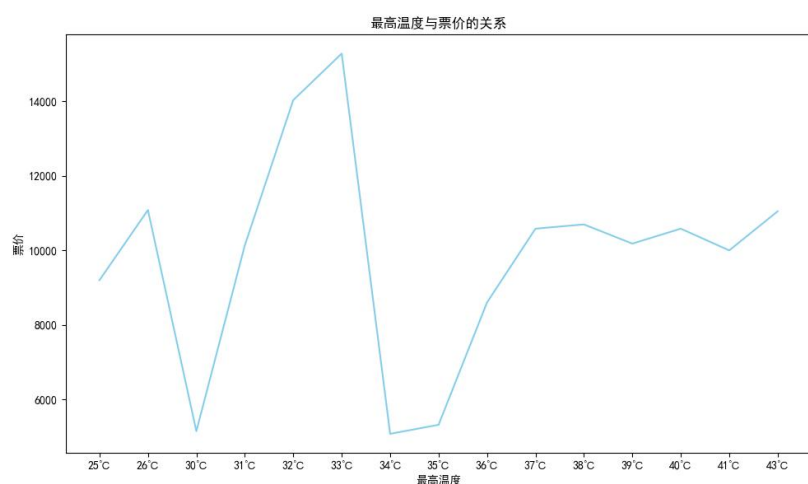


图 10 最高温度与票价

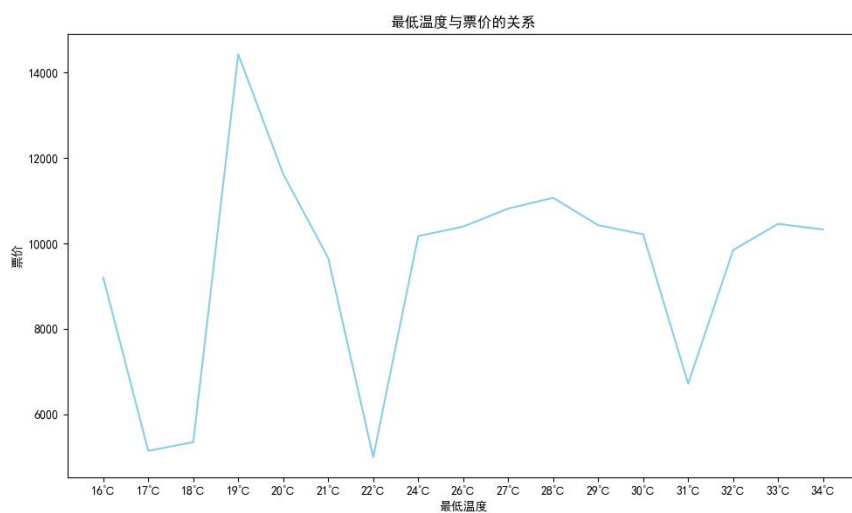


图 11 最低温度与票价

(8) 风向与票价

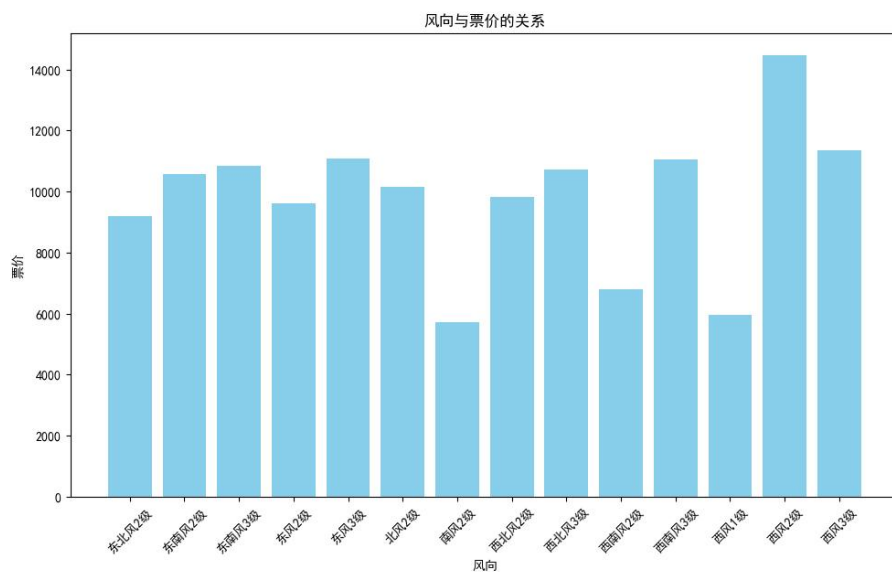


图 12 风向与票价

(9) 降水量与票价

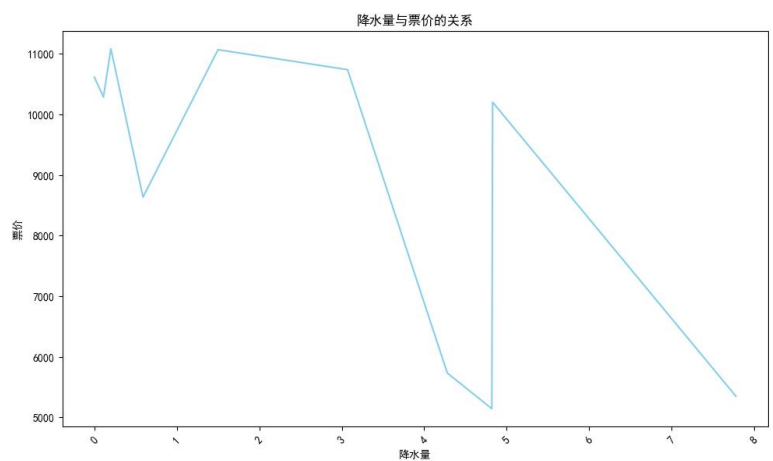


图 13 降水量与票价

(10) 天气与票价及其分布情况

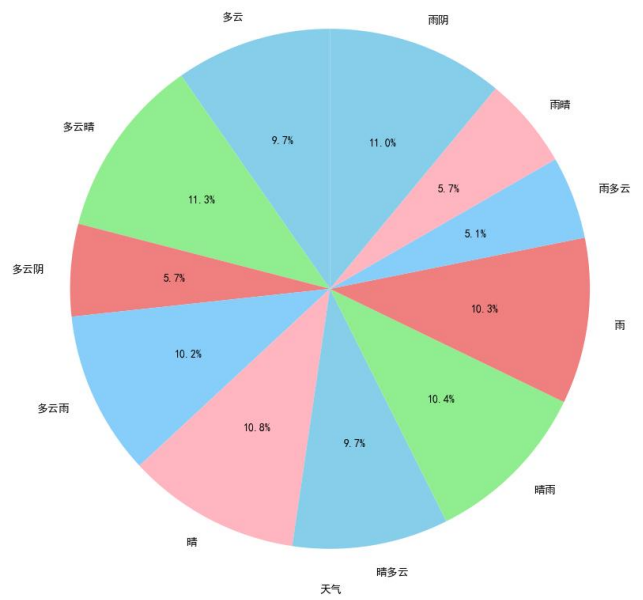


图 14 天气分布情况

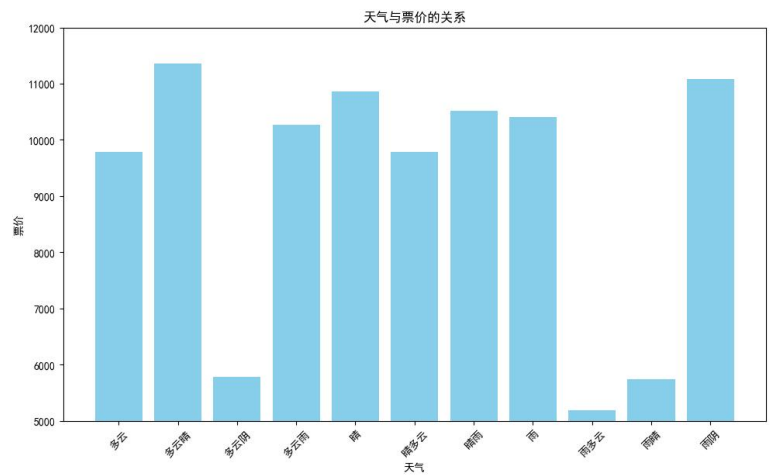


图 15 天气与票价

然后通过 `scipy` 库计算各个影响因素与票价的斯皮尔曼相关系数。较高相关性的几个因素如下表所示：

表 1 几个典型影响因素的斯皮尔曼相关系数

影响因素	斯皮尔曼相关系数
航空公司	0.1907442704225672
出发时间	0.11625763547281855
停站数	0.16117754787703584
风向	0.14803283876592255

1.2.2 附加功能（flask 框架 web 网页与可视化大屏）

我通过 flask 框架构建了前后端分离的 web 应用，web 应用的主要功能如下所示：

☆网站主页：如下图所示，网站的主页包含状态栏、网站介绍。

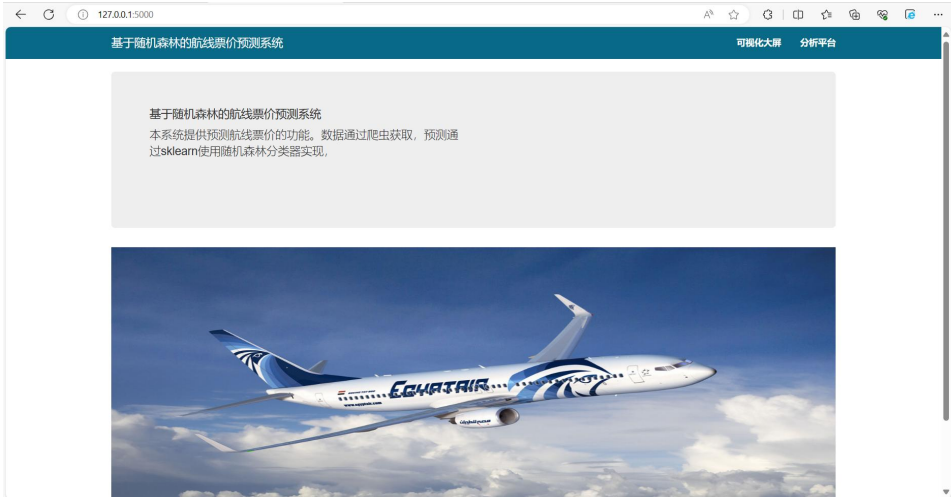


图 16 网站主页

☆可视化大屏：可视化大屏集中展示了数据分析中的部分内容。



图 17 可视化大屏界面

☆预测查询平台：

进入预测平台后，首先点击“浏览文件”，选择需要预测的航次信息并上传，如下图所示：

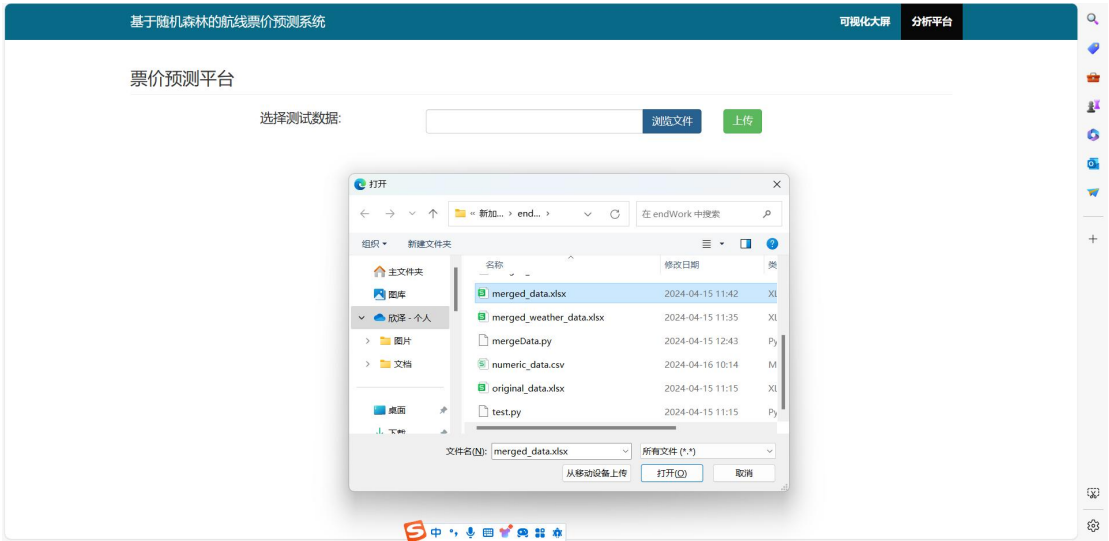


图 18 预测平台上传文件

等待数秒后，显示所需要预测的航线信息，主要包含预测票价、航空公司、出发地、目的地、路线、起飞时间、到达时间、途径时间、停留地数、提供服务、最高温度、最低温度、AQI、风向、降水量、天气等。

界面信息如下图所示：

基于随机森林的航线票价预测系统														
可视化大屏 分析平台														
票价预测平台														
选择测试数据: C:\fakepath\merged_data.xlsx														
预测票价	航空公司	日期	出发地	目的地	路线	起飞时间	到达时间	途径时间	停留地数	提供服务	最高温度	最低温度	AQI	风向 降水量 天气
13882.0	Jet Airways	06-09	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	41°C	33°C	0	西风 3级 0.11 晴雨
8443.33	Multiple carriers	05-27	Delhi	Cochin	DEL → BOM → COK	11:25	19:15	7h 50m	1 stop	No info	37°C	24°C	0	东南风 3级 0.0 雨
8960.49	Air India	06-01	Delhi	Cochin	DEL → BLR → COK	09:45	23:00	13h 15m	1 stop	No info	37°C	26°C	0	西北风 3级 3.07 晴雨
13973.7425	Air India	03-03	Delhi	Cochin	DEL → AMD → BOM → COK	16:40	19:15 04 Mar	26h 35m	2 stops	No info	33°C	20°C	0	西风 2级 0.0 多云 晴
5739.29	SpiceJet	04-15	Delhi	Cochin	DEL → PNQ → COK	08:45	13:15	4h 30m	1 stop	No info	39°C	29°C	0	西风 1级 0.0 晴
10262.0	Jet Airways	06-12	Delhi	Cochin	DEL → BOM → COK	14:00	12:35 13 Jun	22h 35m	1 stop	In-flight meal not included	41°C	34°C	0	西北风 3级 0.0 晴

图 19 预测平台显示结果

1.3 技术路线

技术路线图如下图所示：

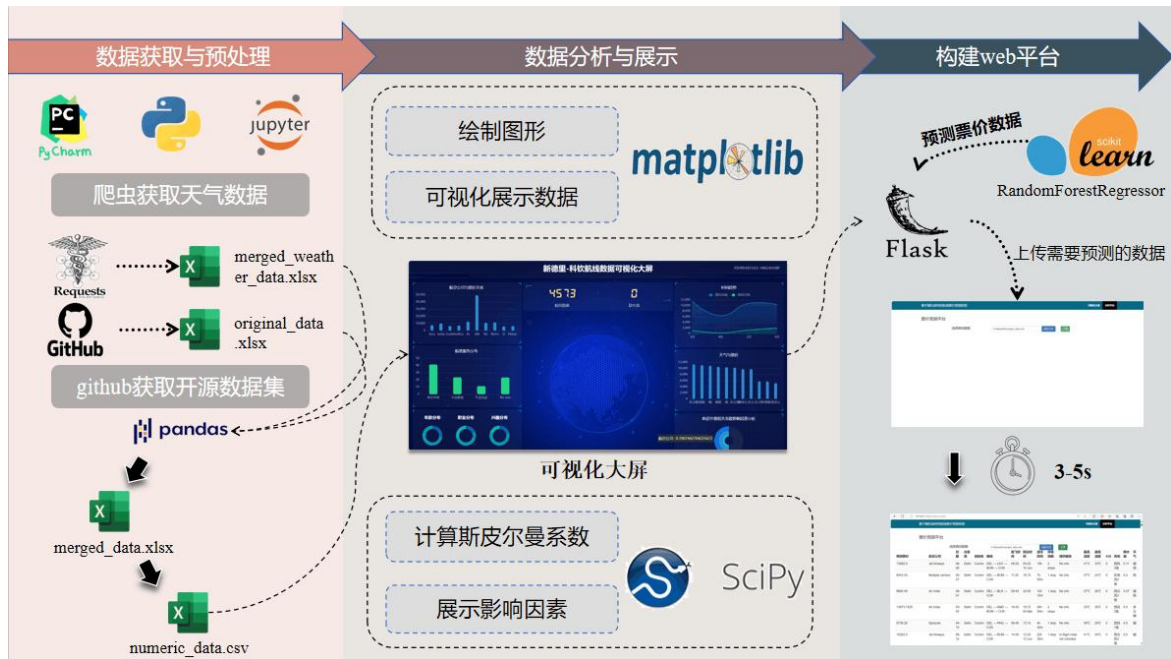


图 20 技术路线图

本项目技术路线主要包含三部分：数据获取与预处理、数据分析与展示、构建 web 平台，使用到了 Python 中的 requests、pandas、matplotlib、scipy、sklearn、flask 等库。

数据获取与预处理部分中，在 endWork/getWeatherData.py 中，我通过 requests 库进行了天气数据的爬取，并通过正则表达式技术匹配到了天气数据，利用 openpyxl 写入到 merged_weather_data.xlsx 中。在 endWork/mergeData.py 中通过 pandas 中的 merge 函数将原始从 github 中获得的数据集 original_data.xlsx 合并为新的总数据集 merged_data.xlsx，再通过 sklearn 将其中的字符串数据转为数字类型，得到 numeric_data.xlsx。

数据分析与展示部分中，在 endWork/dataAnylys.ipynb 中，我通过 pandas 读取了 merged_data.xlsx 并且通过 matplotlib 绘制了航空公司与票价的关系、日期与均价的关系、路线与票价的关系、出发时间与票价关系、持续时间与票价的关系、航班服务与票价的关系及分布情况、最高温度与票价的关系、最低温度与票价的关系、风向与票价的关系、降水量与票价的关系、风向与票价的关系、天气与票价的关系与分布情况等图形，并且使用 scipy 库计算了所有影响因素与票价的斯皮尔曼相关系数，衡量他们与票价的相关性。最终我选取了部分内容在可视化大屏中做展示。

构建 web 平台中，在 endWork/web/web/config.py 中设置了相关配置，在 endWork/web/web/data.py 构造了两个类，作用是为可视化大屏提供数据，在 endWork/web/web/app.py 中先使用 sklearn 框架预测用户上传的数据，并且使用 flask 框架构建 web 应用。其中 endWork/web/web/templates/index.html、endWork/web/web/templates/green.html、endWork/web/web/templates/layout.html、endWork/web/web/templates/model_predict.html 是前端模板，分别为主页界面、可视化大屏界面、状态栏、预测平台界面。

2. 相关技术

2.1 爬虫

requests 库是 Python 中最常用的 HTTP 请求库之一。它提供了一种简单而又灵活的方式来发送 HTTP 请求，并且易于使用。在本项目中，我是用 requests 库爬取新德里天气网站数据，并通过正则表达式技术匹配到天气数据。

网站 https://www.tianqi24.com/in_new-delhi/history202301.html 中的天气数据存在这样的格式下：

```
<li>
<div>01-01</div>
<div>多云</div>
<div class="red">21℃</div>
<div class="green">12℃</div>
<div>0</div>
<div>西北风 1 级</div>
<div>0</div>
</li>
```

可以使用以下正则表达式匹配天气数据：

```
<div>(\d{2}-\d{2})</div>\s*<div>\s*(.*?)\s*(?:<span>\&nbsp;\&nbsp;/\&nbsp;\&nbsp;(.*)?</span>)?\s*</div>\s*<div class="red">(\d+ °C)</div>\s*<div class="green">(\d+ °C)</div>\s*<div>(\d+)</div>\s*<div>(.*?)</div>\s*<div>([\d.]+)</div>
```

2.2 Pandas 操作数据

Pandas 是一个强大的数据分析和处理工具，它提供了丰富的数据结构和函数，能够简化数据的导入、清洗、转换和分析过程。它特别擅长处理结构化数据，例如 Excel 表格、CSV 文件等。

如 read_excel 函数可以读取 excel 文件、head 函数可以预览前 5 行数据、mean 可以求列均值、sort_values 自动排序等。

2.3 Matplotlib 绘制图形

Matplotlib 是一个用于创建可视化图形的 Python 库，它可以创建各种类型的图表，包括线图、散点图、柱状图、饼图等等。

2.4 Sklearn 构建机器学习模型

Scikit-learn（简称 Sklearn）是一个用于机器学习的 Python 库，它包含了大量用于分类、回归、聚类、降维、模型选择和预处理等任务的工具和算法。

Sklearn 常用的函数如下：

划分数据集：

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

模型训练：

```
clf.fit(X_train, y_train)
```

模型预测：

```
y_pred = clf.predict(X_test)
```

利用以上函数，我们可以创建一个随机森林模型用于航线票价预测。

2.5 Flask 框架构建 web 平台

Flask 是一个轻量级的 Python Web 框架，它简单而灵活，适用于构建各种类型的 Web 应用程序，从简单的静态网站到复杂的 Web 应用都可以使用 Flask 来实现。

以下是一段常用的 flask 框架代码示例：

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

在上面的代码中，`@app.route('/')` 装饰器将 URL 路由 / 和视图函数 `index()` 关联起来，当访问根路径时会调用该视图函数返回 `Hello, World!`。

3. 编码实现

3.1 爬虫爬取天气

```
import requests
import re
from openpyxl import Workbook

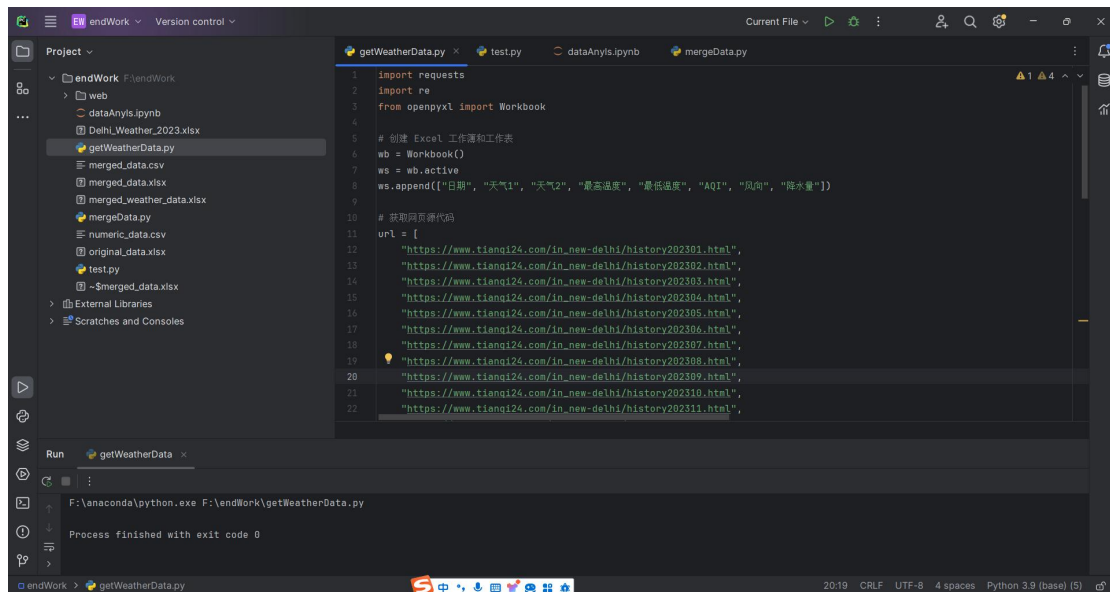
# 获取网页源代码
url = [
    "https://www.tianqi24.com/in_new-delhi/history202301.html",
    "https://www.tianqi24.com/in_new-delhi/history202302.html",
    "https://www.tianqi24.com/in_new-delhi/history202303.html",
    "https://www.tianqi24.com/in_new-delhi/history202304.html",
    "https://www.tianqi24.com/in_new-delhi/history202305.html",
    "https://www.tianqi24.com/in_new-delhi/history202306.html",
    "https://www.tianqi24.com/in_new-delhi/history202307.html",
    "https://www.tianqi24.com/in_new-delhi/history202308.html",
    "https://www.tianqi24.com/in_new-delhi/history202309.html",
    "https://www.tianqi24.com/in_new-delhi/history202310.html",
    "https://www.tianqi24.com/in_new-delhi/history202311.html",
    "https://www.tianqi24.com/in_new-delhi/history202312.html"
]

for i in range(12):
    response = requests.get(url[i])
    html_content = response.text

    # 定义正则表达式模式
    # pattern = r'<div>(\d{2}-\d{2})</div>\s*<div>\s*(.*?)\s*</div>\s*<div>
class="red">(\d+°C)</div>\s*<div>
class="green">(\d+°C)</div>\s*<div>(\d+)</div>\s*<div>(.*?)</div>\s*<div>([\d.]+)
</div>'
    pattern =
r'<div>(\d{2}-\d{2})</div>\s*<div>\s*(.*?)\s*(?:<span>&nbsp;&nbsp;&nbsp;/&nbsp;&nbsp;&nbsp;(.*?)<
/span>)?\s*</div>\s*<div>
class="red">(\d+°C)</div>\s*<div>
class="green">(\d+°C)</div>\s*<div>(\d+)</div>\s*<div>(.*?)</div>\s*<div>([\d.]+)
</div>'

    # 匹配天气信息
    weather_data = re.findall(pattern, html_content)
```

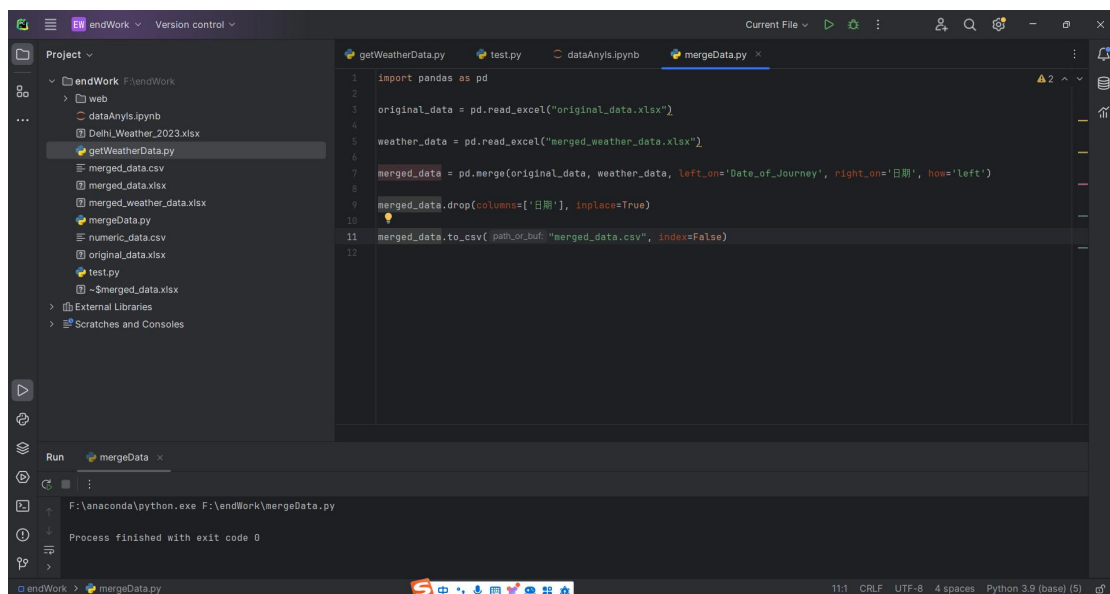
如下图为本段代码成功运行截图：



```
1 import requests
2 import re
3 from openpyxl import Workbook
4
5 # 创建 Excel 工作簿和工作表
6 wb = Workbook()
7 ws = wb.active
8 ws.append(["日期", "天气1", "天气2", "最高温度", "最低温度", "AQI", "风向", "降水量"])
9
10 # 获取网页源代码
11 url = [
12     "https://www.tianqi24.com/in-new-delhi/history202301.html",
13     "https://www.tianqi24.com/in-new-delhi/history202302.html",
14     "https://www.tianqi24.com/in-new-delhi/history202303.html",
15     "https://www.tianqi24.com/in-new-delhi/history202304.html",
16     "https://www.tianqi24.com/in-new-delhi/history202305.html",
17     "https://www.tianqi24.com/in-new-delhi/history202306.html",
18     "https://www.tianqi24.com/in-new-delhi/history202307.html",
19     "https://www.tianqi24.com/in-new-delhi/history202308.html",
20     "https://www.tianqi24.com/in-new-delhi/history202309.html",
21     "https://www.tianqi24.com/in-new-delhi/history202310.html",
22     "https://www.tianqi24.com/in-new-delhi/history202311.html",
```

3.2 pandas 读取数据、合并数据

```
import pandas as pd
original_data = pd.read_excel("original_data.xlsx")
weather_data = pd.read_excel("merged_weather_data.xlsx")
merged_data = pd.merge(original_data, weather_data, left_on='Date_of_Journey',
right_on='日期', how='left')
merged_data.drop(columns=['日期'], inplace=True)
merged_data.to_csv("merged_data.csv", index=False)
```



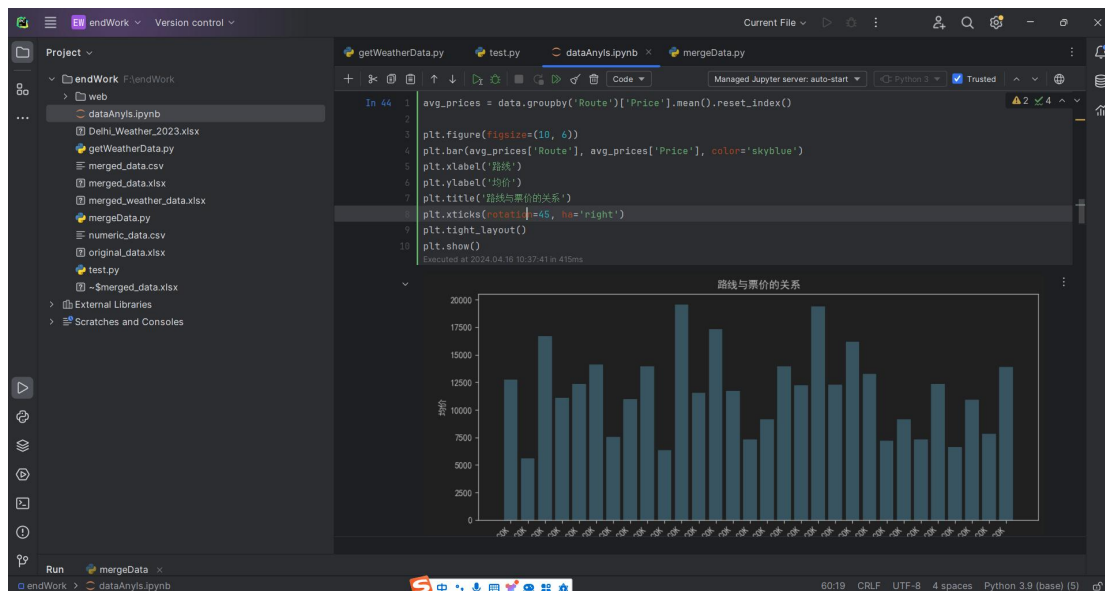
```
1 import pandas as pd
2
3 original_data = pd.read_excel("original_data.xlsx")
4
5 weather_data = pd.read_excel("merged_weather_data.xlsx")
6
7 merged_data = pd.merge(original_data, weather_data, left_on='Date_of_Journey', right_on='日期', how='left')
8
9 merged_data.drop(columns=['日期'], inplace=True)
10
11 merged_data.to_csv(path_or_buf="merged_data.csv", index=False)
12
```

3.3 matplotlib 绘制图形

以路线与票价的关系为例：

```
avg_prices = data.groupby('Route')['Price'].mean().reset_index()
```

```
plt.figure(figsize=(10, 6))
plt.bar(avg_prices['Route'], avg_prices['Price'], color='skyblue')
plt.xlabel('路线')
plt.ylabel('均价')
plt.title('路线与票价的关系')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



3.4 sklearn 构建机器学习模型与 flask 构建 web 平台

```
import pandas as pd
from flask import Flask, render_template, jsonify, request
from data import *
```

```
app = Flask(__name__)
app.config.from_object('config')
# 读取数据集
train_df = pd.read_excel("F:\\endWork\\merged_data.xlsx")
# F:\\endWork\\merged_data.xlsx
# ----- html render -----
@app.route('/')
def index():
```

```

        return render_template('index.html')
@app.route('/people')
def people():
    data = SourceData()
    return render_template('green.html', form=data, title=data.title)
@app.route('/model_predict')
def model_predict():
    return render_template('model_predict.html')
# ----- ajax restful api -----
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
# 特征选择
features = ['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
            'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
            'Additional_Info', '最高温度', '最低温度', 'AQI', '风向', '降水量', '天气']
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
for column in train_df.columns:
    if train_df[column].dtype == 'object':
        train_df[column] = label_encoder.fit_transform(train_df[column])
X = train_df[features]
y = train_df['Price']

# 划分训练集和测试集
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.1,
random_state=42)

# 使用随机森林回归模型
rf_model = RandomForestRegressor()

# 模型训练
rf_model.fit(X_train, y_train)

# 预测并评估模型
y_pred = rf_model.predict(X_valid)
mse = mean_squared_error(y_valid, y_pred)
print("Mean Squared Error:", mse)

```



```

@app.route('/submit_and_predict', methods=['POST'])
def submit_and_predict():
    # 获取上传的文件
    test_file = request.files['file']
    filename = test_file.filename
    # 保存上传的文件
    test_file_path = './data/{}'.format(filename)
    test_file.save(test_file_path)

    # 读取测试集数据
    test_data = pd.read_excel(test_file_path)
    original_test_data = test_data

    # 特征选择
    X_test = test_data[features]

    for column in X_test.columns:
        if X_test[column].dtype == 'object':
            X_test[column] = label_encoder.fit_transform(X_test[column])

    print('-----X_test-----')
    print(X_test)

    # 模型预测
    print('-----test1-----')
    y_pred = rf_model.predict(X_test)

    header = "
    cols = [
        '预测票价', '航空公司', '日期', '出发地', '目的地', '路线',
        '起飞时间', '到达时间', '途径时间', '停留地数',
        '提供服务', '最高温度', '最低温度', 'AQI', '风向', '降水量', '天气'
    ]

    for col in cols:
        if col == '预测':
            header += '<th style="color: red">' + col + '</th>'
        else:
            header += '<th>' + col + '</th>'
    print('-----test2-----')

    rows = "
    test_data_original = pd.read_excel(test_file_path)
    X_test_original = test_data_original[features]

```

```

for i in range(len(y_pred)):
    row = '<tr><td>{}'.format(y_pred[i])
    #print(X_test)
    for d in X_test_orignal.iloc[i]:
        # print("-----",d)
        row += '<td>' + str(d) + '</td>'
    row += '</tr>'
    rows += row
print('-----test3-----')

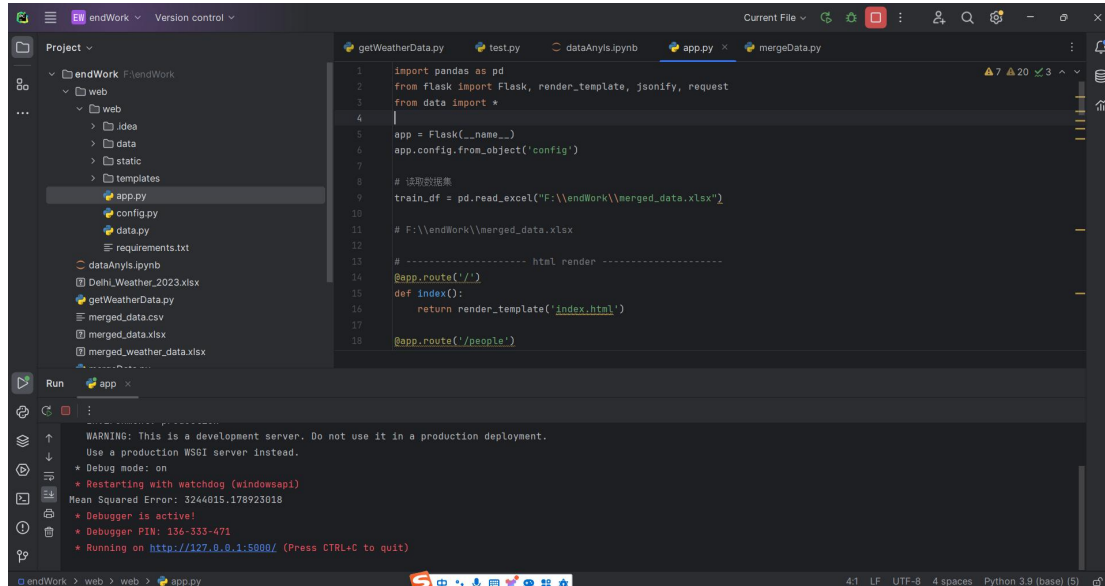
return jsonify({
    'success': True,
    'acc': 0,
    'precision': 0,
    'recall': 0,
    'header': header,
    'rows': rows
})

```

```

if __name__ == "__main__":
    app.run(host='127.0.0.1')

```



4. 遇到的主要问题及解决方法

①数据问题：航班的票价是与某些天气因素相关的，但查到的数据集内不包含有天气数据，因此我通过爬虫技术爬取了新德里地区的天气数据，将数据集进行合并得到了可供研究的完整数据集。

②框架学习问题：由于我没有 flask 框架构建 web 应用的经验，因此我选择学习了 github 上一些 flask 模板的代码，并对 flask 框架做了初步了解，最终在模板的基础上修改，得到我自己想要的网站界面及功能。

③模型选择问题：sklearn 库提供了多种机器学习回归模型，我对于航线票价预测领域的常用模型不了解，最终通过查阅文献确定了随机森林模型的使用可以得到较为准确的结果。

5. 总结与展望

在本次项目中，我设计并实现了基于机器学习的航班票价数据分析及可视化平台。通过这个项目，我获益良多。

首先，我深入了解了机器学习在实际数据分析中的应用。通过数据的获取、清洗、特征工程、模型选择和训练等步骤，我对机器学习算法的原理和实践有了更深入的理解，同时也掌握了一些常用的数据分析和机器学习工具和技术。

其次，我学会了如何设计和实现一个完整的数据分析和可视化平台。从需求分析、系统设计、编码实现到测试部署，我独立完成了项目的各个环节，对软件工程的了解进一步加深。

另外，我也意识到在实际项目中，面临的挑战和问题常常是多样化和复杂化的。在项目的过程中，我遇到了数据获取困难、模型性能不佳等问题，但通过不懈的努力，最终都得以解决，并取得了一定的成果。

展望未来，我认为我设计的平台还有许多改进和完善的空间。首先，可以进一步优化模型的性能和预测精度，通过引入更多的特征工程技术和模型调优方法，提高模型的泛化能力和适用性。其次，可以考虑引入更多的数据源和数据类型，如天气数据、航班历史数据等，丰富数据的维度和内容，提高分析的深度和广度。