《Python 高级应用》

大作业实验报告

项目名称 ： TCMDF：清心医谷 traditional chinese medicine data fusion 中医药智慧分析

姓　　名　　：　吴泽同

学　　号　　：　084622109

班　　级　　：　计算机类 221

成　　绩　　：

人工智能与信息技术学院

南京中医药大学

项目要求：

数据分析和模型训练：

读取 Excel 文件中的数据，并进行特征提取和标签处理。

使用线性回归模型进行训练，并输出模型评估指标（如均方误差和解释方差分）。

词云生成：

从用户输入的文本中生成词云，使用 jieba 进行分词处理。

可以加载自定义的鸽子图片，并将词云生成在图片上。

时间序列预测：

从上传的 CSV 文件中读取时间序列数据，假设是中医药销售量和售价数据。

使用 SARIMAX 模型进行时间序列预测，并输出未来时间点的预测值。

神经网络可视化：

从上传的 CSV 文件中读取数据，进行特征和标签提取。

将字符串标签转换为数字标签，并创建 3D 图像展示数据分布。

采用的技术

技术方面，项目采用了以下库和技术：
Python：作为开发语言。
Flask：用于构建 Web 应用，实现路由、模板渲染等功能。
pandas：用于数据读取和处理。
scikit-learn：提供线性回归模型和数据预处理功能。
statsmodels：提供时间序列分析的 SARIMAX 模型。
jieba：用于中文文本的分词处理。
wordcloud：用于生成词云图像。
matplotlib：用于绘制图表和可视化。
keras：用于神经网络的建模和训练。
进一步延申可以涉及到数据库，可以使用 SQLite 或 MySQL，并结合 SQLAlchemy 进行数据存取操作。

关键代码及相应运行结果

(请将项目从前到后的关键代码贴出，并加上相应的运行结果截图。)

```
import os
from datetime import datetime, timedelta
```

```
from PIL import Image, ImageOps

from flask import Flask, render_template, current_app
from keras.layers import Dense
from keras.models import Sequential
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from statsmodels.tsa.statespace.sarimax import SARIMAX

app = Flask(__name__)
app.config['STATIC_FOLDER'] = 'static'  # 设置静态文件夹路径

app.config['CUSTOM_CSS'] = 'custom.css'
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# 从 Excel 文件中读取数据
data = pd.read_excel('ex.xlsx')

# 提取特征和标签
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 训练模型
model = LinearRegression()
model.fit(X_train, y_train)

# 在测试集上进行预测
y_pred = model.predict(X_test)

# 打印模型评估指标 (可根据具体需求进行修改)
print("模型评估指标:")
print("均方误差 (MSE) :", mean_squared_error(y_test, y_pred))
print("解释方差分 (R^2) :", r2_score(y_test, y_pred))



@app.route('/')
def index():
```

```
    custom_css = app.config['CUSTOM_CSS']
    return render_template('index3.html', custom_css=custom_css)

import jieba
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties


@app.route('/generate_wordcloud', methods=['POST'])
def generate_wordcloud():
    text = request.form['text']
    word_list = jieba.lcut(text)
    processed_text = " ".join(word_list)  # 将分词结果用空格连接起来

    font_path = 'C:/Windows/Fonts/SIMLI.TTF'  # 替换成你的字体文件路径
    font_prop = FontProperties(fname=font_path, size=16)  # 设置字体属性

    # 加载鸽子图片并转化为 numpy 数组
    pigeon_mask                                                    =
np.array(ImageOps.invert(Image.open("pigeon.png").convert('RGB')))

    wordcloud = WordCloud(font_path=font_path, font_step=2, mask=pigeon_mask,
background_color='white').generate(processed_text)

    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.savefig(os.path.join(app.config['STATIC_FOLDER'], 'wordcloud.png'))

    return render_template('index3.html', image='wordcloud.png')

from flask import render_template, request
import pandas as pd
import numpy as np


@app.route('/time_series_prediction', methods=['POST'])
def time_series_prediction():
    # 从上传的 CSV 文件中读取数据
    file = request.files['file']
    data = pd.read_csv(file, encoding='GBK')

    # 提取时间序列数据（假设中医药销售量和售价作为时间序列数据）
    sales_data = data['销售量'].values
```

```
    price_data = data['售价'].values

    # 创建 SARIMAX 模型并进行训练
    model = SARIMAX(sales_data, order=(1, 0, 0), seasonal_order=(0, 0, 0, 0))
    model_fit = model.fit()

    # 进行未来时间点的预测
    forecast_sales = model_fit.forecast(steps=10)

    # 计算未来时间点的日期
    last_date = datetime.strptime(data['日期'].values[-1], '%Y-%m-%d')
    future_dates = [last_date + timedelta(days=i) for i in range(1, 11)]

    # 输出日期、销售量和售价的预测值
    output_str = "<p>Future Predictions:</p><ul>"
    for date, sales, price in zip(future_dates, forecast_sales, price_data):
        output_str += f"<li>{date.strftime('%Y-%m-%d')}: 销售量: {sales:.2f},
售价: {price:.2f}</li>"
    output_str += "</ul>"
    return render_template('index3.html', result=output_str)
@app.route('/neural_network_3d_plot', methods=['GET', 'POST'])
def neural_network_3d_plot():
    if request.method == 'POST':
        file = request.files['file']
        if file and file.filename.endswith('.csv'):
            # 从上传的 CSV 文件中读取数据
            data = pd.read_csv(file)

            # 提取特征和标签
            X = data.iloc[:, :-1].values
            y = data.iloc[:, -1].values

            # 将字符串标签转换为数字标签
            label_encoder = LabelEncoder()
            y_encoded = label_encoder.fit_transform(y)

            # 创建 3D 图像
            fig = plt.figure()
            ax = fig.add_subplot(111, projection='3d')
            ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y_encoded)
            ax.set_xlabel('X1')
            ax.set_ylabel('X2')
            ax.set_zlabel('X3')
```

```
            # 保存图片到静态文件夹
            plot_path = os.path.join(app.config['STATIC_FOLDER'],
'3d_plot.png')
            plt.savefig(plot_path)

            return render_template('index3.html',
three_d_plot_image='3d_plot.png')  # 返回保存图片的名称

    return render_template('index3.html')  # 如果未上传文件，则仅渲染模板
if __name__ == '__main__':
    app.run(debug=True)
```

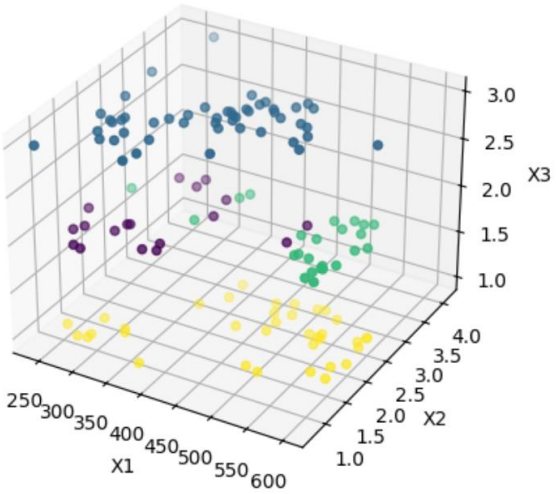
flask.debughelpers.DebugFilesKe × Flask App ×

127.0.0.1:5000/neural_network_3d_plot

Enter text: 论文词云——生成词云图

选择文件 drug_data.csv 神妙图谱——上传文件并生成3D神经网络图像

选择CSV文件: 选择文件 未选择文件 时序卜卦——中药销售量和售价的时序分析



flask.debughelpers.DebugFilesKe × Flask App ×

127.0.0.1:5000/time_series_prediction

Enter text: 论文词云——生成词云图

选择文件 未选择文件 神妙图谱——上传文件并生成3D神经网络图像

选择CSV文件: 选择文件 weather_data.csv 时序卜卦——中药销售量和售价的时序分析

Future Predictions:

- 2023-02-21: 销售量: 1335.99, 售价: 11.00
- 2023-02-22: 销售量: 1322.12, 售价: 12.00
- 2023-02-23: 销售量: 1308.39, 售价: 10.00
- 2023-02-24: 销售量: 1294.81, 售价: 14.00
- 2023-02-25: 销售量: 1281.37, 售价: 11.00
- 2023-02-26: 销售量: 1268.06, 售价: 12.00
- 2023-02-27: 销售量: 1254.90, 售价: 13.00
- 2023-02-28: 销售量: 1241.87, 售价: 10.00
- 2023-03-01: 销售量: 1228.98, 售价: 13.00
- 2023-03-02: 销售量: 1216.22, 售价: 13.00