20231211

TCMDF：清心医谷

traditional chinese medicine data fusion中医药智慧分析

084622109吴泽同

目录

- 1. 生成词云
- 2. 时间序列预测
- 3. 神经网络三维可视化
- ？ . Web应用

1. 生成词云

1. 生成词云

- 分词处理
- 生成词云图像
- 展示词云图像


```python
@app.route( rule: '/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

    # 加载图片并转化为 numpy 数组
    pigeon_mask = np.array(ImageOps.invert(Image.open("china.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( template_name_or_list: 'index3.html', image='wordcloud.png')
```

1. 生成词云

- 对输入文本进行分词处理，使用jieba.lcut函数将文本分割成词语列表。然后，使用空格连接分词结果，并使用WordCloud类来生成词云图像。在生成词云之前，可以设置字体属性，如字体文件路径和大小。最后，使用matplotlib.pyplot库展示词云图像。

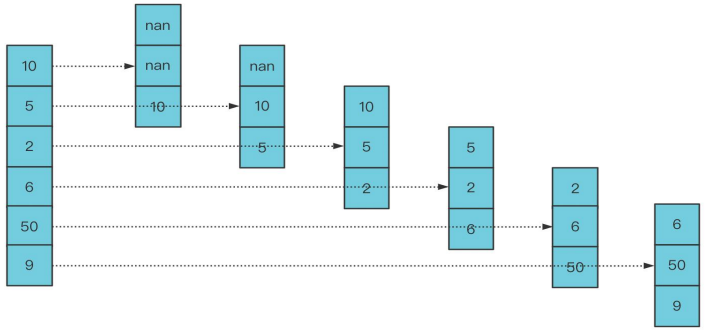
jieba库常用函数

函数	描述
jieba.lcut(s)	精确模式，返回一个列表类型的分词结果 >>>jieba.lcut("中国是一个伟大的国家") ['中国', '是', '一个', '伟大', '的', '国家']
jieba.lcut(s, cut_all=True)	全模式，返回一个列表类型的分词结果，存在冗余 >>>jieba.lcut("中国是一个伟大的国家",cut_all=True) ['中国', '国是', '一个', '伟大', '的', '国家']

2. 时间序列预测

2. 时间序列预测

- - 从CSV文件中读取时间序列数据
- - 创建SARIMAX模型并进行训练
- - 进行未来时间点的预测




```python
@app.route( rule: '/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], __format: '%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( template_name_or_list: 'index3.html', result=output_str)
```

2. 时间序列预测

- 首先，通过pd.read_csv函数从CSV文件中读取时间序列数据。然后，使用SARIMAX类创建SARIMA模型，并使用fit方法进行训练。最后，使用forecast方法进行未来时间点的预测。

定义

- 时间序列：将预测对象按照时间顺序排列而成的序列。
- 时序预测：根据时序过去的变化规律，推测今后趋势。

时间序列的变化形式

- 长期趋势变动 T_t
- 季节变动 S_t
- 循环变动 C_t
- 不规则变动 R_t

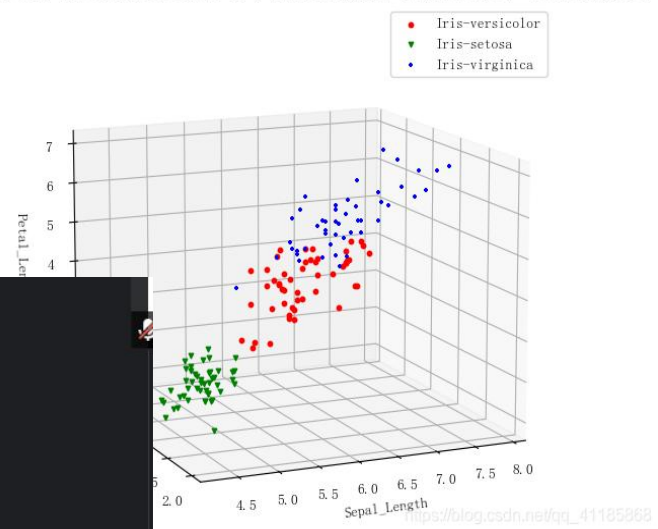
模型

- 加法模型
- 乘法模型
- 混合模型

3. 神经网络三维可视化

3. 神经网络三维可视化

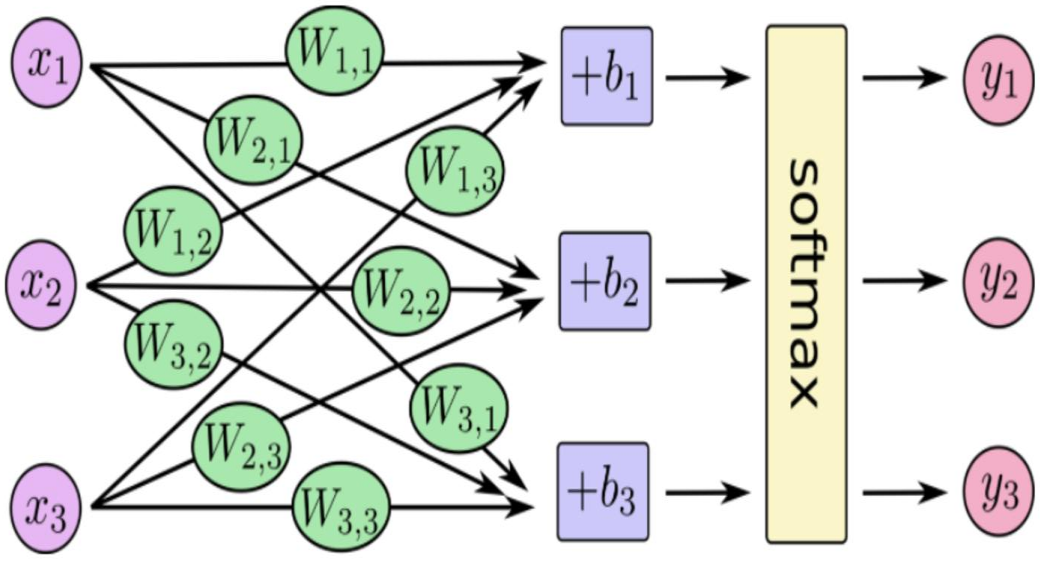
- - 生成随机数据
- - 构建神经网络模型
- - 训练模型
- - 生成三维散点图




```python
@app.route( rule: '/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( template_name_or_list: '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
```

3. 神经网络三维可视化

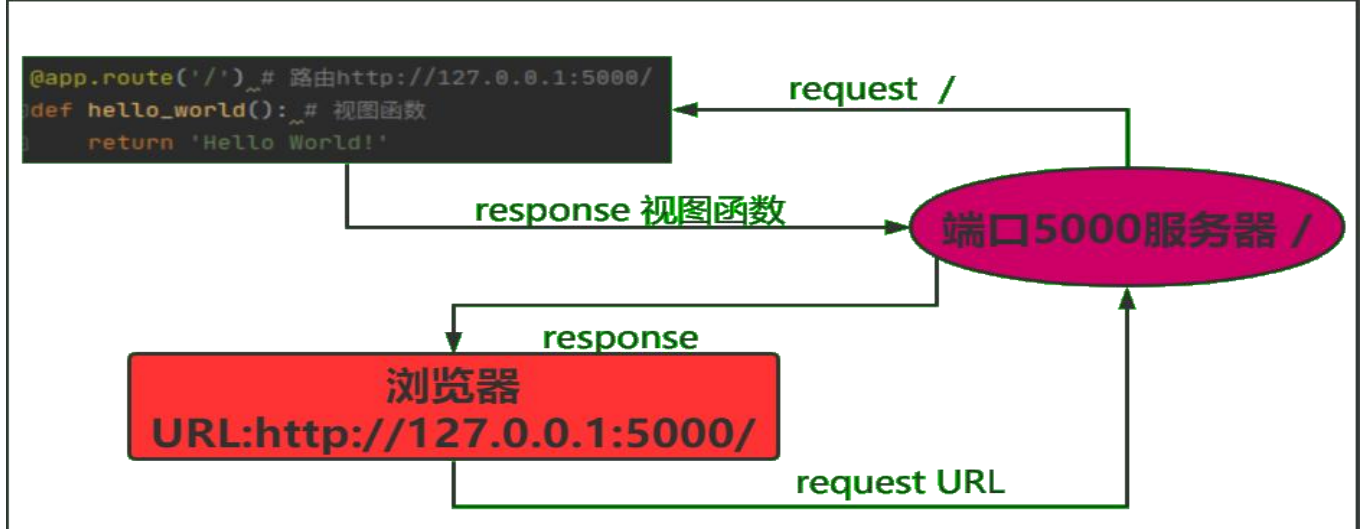
- 这部分代码生成了随机数据，使用Sequential类创建神经网络模型，并使用Dense类添加隐藏层和输出层。然后，使用compile方法编译模型，指定损失函数、优化器和评估指标。接着，使用fit方法对模型进行训练。最后，使用matplotlib.pyplot和mpl_toolkits.mplot3d库生成三维散点图。



4. Web应用

4. Web应用

- - 使用Flask构建Web应用
- - 设置静态文件夹路径
- - 定义路由和视图函数
- - 运行应用




```python
@app.route('/')
def index():
    # 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))
    custom_css = app.config['CUSTOM_CSS']
    return render_template( template_name_or_list: 'index3.html', custom_css=custom_css)
```

4. Web应用

- 使用Flask框架构建Web应用。首先，通过Flask类创建应用实例。然后，设置静态文件夹路径，通过app.config['STATIC_FOLDER']来指定静态文件夹的路径。接着，使用@app.route装饰器定义路由和相应的视图函数。最后，通过app.run方法运行应用，并设置debug=True来开启调试模式。