一．选题描述：

本课程设计旨在构建性能优越的葡萄酒品级预测模型，通过对数据进⾏分析和可视化来评价各个特征（寻找评价指标，定量地判断单个或多个联合特征，对分类结果的贡献，并以此确定需要⽤于分类的特征⼦集）、使用传统机器学习和深度学习构建分类模型（回归模型、决策树模型等），完成回归任务，并使⽤F1 Score，在验证集上检验⾃⼰的模型，实现葡萄酒品质打分。数据集采用csv格式，⾸⾏为标题⾏，⽽其余⾏是由"\t"分割的数据字段，包含5574条英⽂短消息。⽤前4000⾏作为训练集，其余⽤作验证集。

二．设计方案：

1. 对数据进⾏分析和可视化，评价各个特征：

寻找评价指标，定量地判断单个特征，对分类结果的贡献

• 计算协方差矩阵与分类结果的相关系数

• 通过数据可视化，直观地展⽰出指标与分类结果的关系

• 在以上结果的基础上，确定需要⽤于分类的特征⼦集（选择时需要注意，特征并不是越多越好，与任务标签相关度低的特征，可能会成为噪声，⼲扰模型的决策）。

机器学习和深度学习都是用于构建模型的方法，它们在一些方面有所不同。但大致步骤均如下：数据预处理，特征提取，模型选择，训练模型，模型检验。

2. 构建基于传统机器学习⽅法的分类模型：

基于对特征的分析和评价，选取⼀些特征，使⽤回归模型和决策树模型，实现分类模型。使⽤F1 Score，在验证集上检验此模型。

3. 构建基于神经⽹络的分类模型：

利用神经网络（CNN、RNN、Transformer等）构建分类模型，在验证集上检验模型。在第一步的基础上使用各种方式来优化并提升分类的准确率。

三．代码模块的功能划分与描述：

1. 数据预处理：读取csv文件，将前5000条作为训练集，剩余的作为验证集。

# 加载数据集，注意sep参数表示更换分隔符

dataset = pd.read\_csv("wine.csv")

# 前4000句作为训练集，4001到末尾作为验证集

train\_dataset = dataset.head(4000)

valid\_dataset = dataset.tail(-4000)

2.数据分析：计算数据集的协方差矩阵，计算不同方法（kendall，spearman等）的相关系数矩阵并绘制条形图，借此反映单个特征对分类结果的贡献并绘制热力图，实现数据可视化。

# 计算数据集的协方差矩阵

df\_corr = train\_dataset.corr()

sns.heatmap(df\_corr, vmax=1, vmin=-1, center=0)

# 绘制热力图，实现数据可视化

plt.show()

print(df\_corr[u'quality'])

df1=df\_corr[u'quality']

# 计算不同方法的相关系数矩阵

df2=train\_dataset.corr(method='kendall')[u'quality']

print(df2)

df3 = train\_dataset.corr(method='spearman')[u'quality']

print(df3)

3.机器学习特征提取：手动设计规则，通过结果选出训练集测试及参数。

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.5, random\_state=42)

print('X\_train.shape={}\n y\_train.shape={}\n X\_test.shape={}\n y\_test.shape={}'.format(X\_train.shape, y\_train.shape,X\_test.shape,y\_test.shape))

4.机器学习模型训练及模型检验：使用训练集进行模型训练并使用验证集进行模型预测，计算F1 Score。

4-1回归模型：（与手算RMSE比较，进行模型检验）  
linreg = LinearRegression() # 建立模型对象

# 训练模型并输出模型参数和权重

model = linreg.fit(X\_train, y\_train)

y\_pred = linreg.predict(X\_test)

F1\_Score检验：

y\_pred = np.around(y\_pred,0).astype(int)

f1\_micro=sklearn.metrics.f1\_score(y\_test, y\_pred, average='micro')

print(f'线性回归：F1\_score\_micro:{f1\_micro}')

4-2决策树模型:

clf=DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

y\_pred=clf.predict(X\_test)

y\_pred = np.around(y\_pred,0).astype(int)

f2\_micro=sklearn.metrics.f1\_score(y\_test, y\_pred, average='micro')

print(f'决策树：F1\_score\_micro:{f2\_micro}')

5. 深度学习模型

选择神经网络结构：CNN主要用于图像和视频数据的处理，而RNN和Transformer可以更好地处理文本数据。此外，由于数据集中包含的是一个代表quality的整数和其他性质的浮点数，因此考虑使用简单的RNN模型来处理这些数据。

具体来说，可以使用一个简单的RNN模型，例如基于长短时记忆网络(LSTM)的模型，或者一个基于双向LSTM的模型。这些模型可以处理可变长度的输入序列，并且可以通过反向传播算法进行端到端的训练。另外，还可以使用一些预处理技术，例如分词和字符编码，来将输入数据转换为模型可接受的形式。

- 数据预处理：读取csv文件，将前5000条作为训练集，剩余的作为验证集。

- 特征提取：使用神经网络（CNN、RNN、Transformer等）进行特征提取。

- 模型训练：使用训练集进行模型训练。

- 模型预测：使用验证集进行模型预测，计算F1 Score。

四．实现效果：

通过对三种模型的实现和对比，可以得出不同模型的优缺点和适用场景。规则模型简单易懂，但需要手动设计规则，不适用于大规模数据；传统机器学习模型适用于中小规模数据，但需要手动提取特征，对于复杂数据表现不佳；深度学习模型适用于大规模数据，可以自动提取特征，但需要更多的计算资源和时间。通过对比，可以选择最适合实际应用场景的模型，提高诈骗短信检测的准确率和效率。

