H34091071 郭哲豪 # HW 3

一、資料簡介

Wave Dataset，為一個著名的人工波浪數據集（Breiman et al., 1984），也被廣泛用於分析方法。最初，這個波浪數據集有 3 個類別，21 個變量，每個類別的變量是基於兩個或三個波浪形式的隨機凸集合加上噪音生成的。Rakotomalala（2005）通過添加噪音變量和生成更多主題將數據集擴展，並將這個數據集修改為 Binary Response Dataset，只保留了前兩個類別的主題。

目前，這個數據集中有 33,334 個主題，其中 10,000 個主題被收集作為訓練集。這是一個平衡的分類問題，因為兩個類別的主題數量幾乎相同。除了原來的 21 個預測變量外，還添加了 100 個噪音變量，它們與相應的分類問題完全獨立。

二、分析概念說明

針對這 binary classification problem，進行兩階段地分析方法，並同時執行分類及變數選擇，目的是對機器學習模型進行訓練、特徵重要性分析和性能評估，以了解模型對於分類問題的預測能力和特徵的重要性。

1. **載入數據：**使用 arff.loadarff() 函數從 ARFF 文件中載入數據，並將其轉換為 Pandas DataFrame，分為訓練集和測試集。
2. **資料預處理：**對目標變量進行二元分類的編碼，選擇一個類別作為正類。同時，對數據進行標準化、缺失值填補等預處理操作。
3. **定義預測模型：**使用 LogisticRegression、LinearSVC、RandomForest、LinearRegression 模型作為預測模型，設定正則化參數和優化算法。
4. **模型訓練：**將訓練集分為訓練子集和驗證子集，使用訓練子集對模型進行訓練。

5. **特徵重要性分析：**運用 Lasso、Ridge 迴歸、Grafting technique 等 Regularization 方法，通過模型的係數來計算特徵的重要性，並將結果存儲在 DataFrame 中，按照重要性值的降序排序。
6. **預測和評估：**對驗證子集進行預測，並計算模型的評估指標，如準確率、精確度、召回率和 F1-score。

三、分析結果

1. 利用 python 來做此分析，引用函示庫

```
from scipy.io import arff
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

2. 匯入資料集，並將其轉換為 Pandas DataFrame，以及定義訓練集和測試集

```
# Specify the paths of the ARFF files
arff_file1 = '/Users/brady/Desktop/機器學習/wave_2_classes_with_irrelevant_attributes/wav
arff_file2 = '/Users/brady/Desktop/機器學習/wave_2_classes_with_irrelevant_attributes/wav

# Use the arff package to load the ARFF files
data1, meta1 = arff.loadarff(arff_file1)
data2, meta2 = arff.loadarff(arff_file2)

# Convert ARFF data to Pandas DataFrame
X_train = pd.DataFrame.from_records(data1)
X_test = pd.DataFrame.from_records(data2)

# Extract the target variable
y_train = X_train.pop('classe')
y_test = X_test.pop('classe')
```

3. 做資料預處理，對目標變量進行二元分類的編碼

```
# Perform binary classification by selecting one class as positive
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)
```

4. 利用 Logistic Regression Model 做預測，以 lasso 的方法選擇重要變數，C 調整 Regularization 的強度，並開始訓練

```
# Define the prediction model
model = LogisticRegression(penalty='l1', solver='liblinear',C=0.1)  # Stronger regularization

# Split the data into training and test sets
X_train_split, X_test_split, y_train_split, y_test_split = train_test_split(X_train, y_train,
                                                                            test_size=0.2, random_state=42)

# Train the model
model.fit(X_train_split, y_train_split)
```

利用 Linear SVM Model 做預測，以 ridge 迴歸的方法選擇重要變數，並開始訓練

```
# Define the prediction model
model = LinearSVC(penalty='l2')
```

利用 Random Forest Model 做預測及選擇重要變數，並開始訓練

```
# Define the prediction model
model = RandomForestClassifier()
```

利用 Linear Regression Model 做預測，以 Grafting technique 的方法選擇重要變數，並開始訓練

```
# Define the prediction model
model = LinearRegression()
```

```
def chebychev_greedy(X_train, y_train, k):
    selected_features = []  # 存儲選中的特徵的索引
    remaining_features = list(range(X_train.shape[1]))  # 初始化剩餘特徵列表

    # 選擇前k個最重要的特徵
    for _ in range(k):
        best_score = -np.inf
        best_feature = None

        # 遍歷剩餘特徵，找到分數最大的特徵
        for feature in remaining_features:
            # 構建子特徵集合
            selected_features.append(feature)
            X_selected = X_train[:, selected_features]

            # 使用選擇的特徵訓練模型並計算分數（例如準確度）
            model = LinearSVC()
            model.fit(X_selected, y_train)
            score = np.abs(model.coef_).max()  # 使用特徵的分數作為指標

            # 更新最佳特徵和分數
            if score > best_score:
                best_score = score
                best_feature = feature

            # 移除試探性添加的特徵
            selected_features.remove(feature)

        # 將最佳特徵添加到選中特徵列表中
        selected_features.append(best_feature)
        remaining_features.remove(best_feature)

    return selected_features
```

5. 計算特徵的重要性，並將結果存儲在 DataFrame 中，按照重要性值的降序排序印出

```
# Calculate feature importances
feature_importances = model.coef_[0]

# Create a DataFrame to store the feature importances
importance_df = pd.DataFrame({'Feature': X_train.columns, 'Importance': feature_importances})

# Sort the DataFrame by importance values in descending order
importance_df = importance_df.sort_values('Importance', ascending=True)

# Print the important variables
print(importance_df)
```

6. 對驗證子集進行預測，並計算模型的評估指標：準確率、精確度、召回率和 F1-score

```
# Make predictions on the test set
y_pred = model.predict(X_test_split)

# Calculate evaluation metrics (e.g., accuracy, precision, recall, f1-score)
accuracy = accuracy_score(y_test_split, y_pred)
precision = precision_score(y_test_split, y_pred, average='weighted')
recall = recall_score(y_test_split, y_pred, average='weighted')
f1 = f1_score(y_test_split, y_pred, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

7. 輸出結果

```
    Feature  Importance
16      v17   -0.576289
15      v16   -0.499913
17      v18   -0.441406
14      v15   -0.429283
4        v5   -0.313287
..      ...         ...
7        v8    0.255314
11      v12    0.446353
8        v9    0.512827
9       v10    0.549294
10      v11    0.732468

[121 rows x 2 columns]
Accuracy: 0.9215
Precision: 0.921881909871953
Recall: 0.9215
F1-score: 0.9214971543572148
```

以 Logistic Regression Model、lasso 方法為例

輸出結果中的特徵重要性（Feature Importance）列出了每個特徵（Feature）的重要性值（Importance）。這些值代表該特徵對於模型預測的貢獻程度，值越高表示該特徵對於模型的影響越大。

在輸出結果中，特徵重要性按照由小到大的順序排列。例如，第一行的特徵v17的重要性值為 -0.576289，表示這個特徵對於模型預測的貢獻為負向，即該特徵的值越高，模型預測的結果越傾向於負向類別。

Accuracy（準確率）：表示模型在測試集上的正確率，為0.9215，即模型對於測試集中的樣本正確預測的比例。
Precision（精確率）：是在預測為正類別的樣本中，實際為正類別的比例，為0.921881909871953。
Recall（召回率）：是在實際為正類別的樣本中，預測為正類別的比例，也可以解釋為正樣本被正確檢測出來的比例，為0.9215。
F1-score（F1值）：是精確率和召回率的調和平均值，綜合考慮了模型的準確度和召回率，為0.9214971543572148。

這些評估指標可以用來評估模型在測試集上的性能，並衡量模型的預測效果。

LinearSVM

```
   Feature  Importance
16     v17   -0.193402
15     v16   -0.154390
17     v18   -0.142532
14     v15   -0.136996
4       v5   -0.105865
..     ...         ...
44  alea24    0.164466
11     v12    0.169451
8       v9    0.190250
9      v10    0.199916
10     v11    0.260349

[121 rows x 2 columns]
Accuracy: 0.918
Precision: 0.9180516402065608
Recall: 0.918
F1-score: 0.9180028701406368
```

RandomForest

```
   Feature  Importance
41  alea21    0.002225
68  alea48    0.002228
80  alea60    0.002232
28   alea8    0.002258
37  alea17    0.002305
..     ...         ...
15     v16    0.057692
8       v9    0.071495
16     v17    0.090559
9      v10    0.114923
10     v11    0.132050

[121 rows x 2 columns]
Accuracy: 0.9135
Precision: 0.9153196188092352
Recall: 0.9135
F1-score: 0.9134423451455475
```

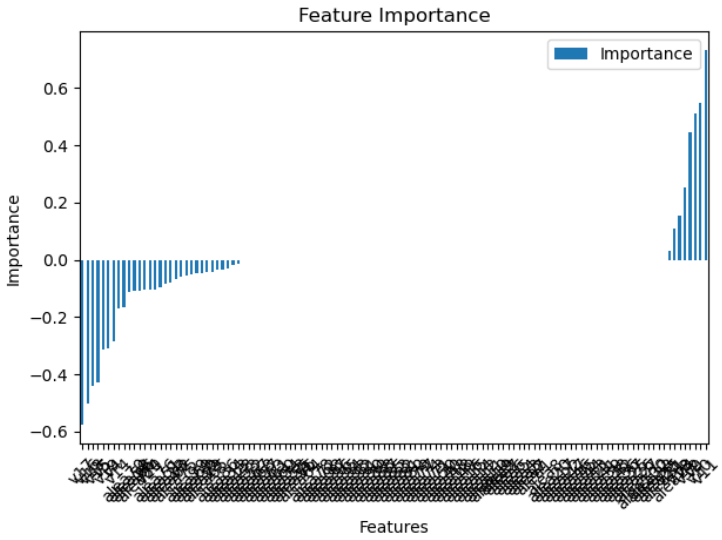
LinearRegression

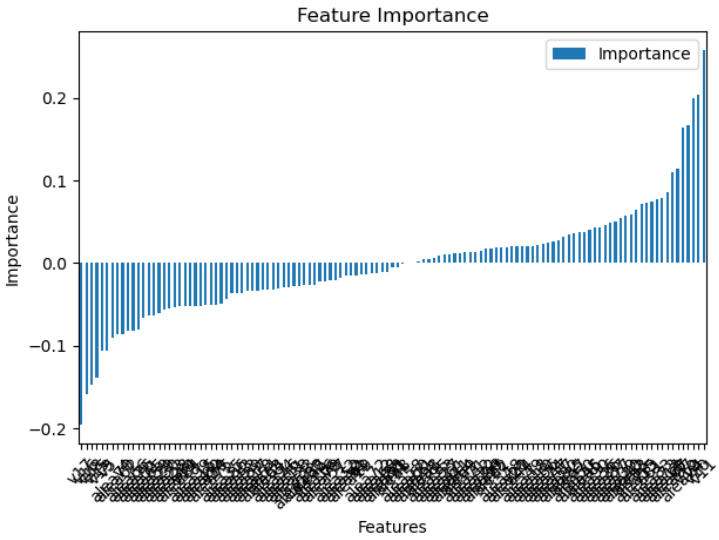
```
   Feature  Importance
60  alea40    0.000000
22   alea2    0.000000
64  alea44    0.000000
24   alea4    0.000000
62  alea42    0.000000
..     ...         ...
15     v16    0.039868
16     v17    0.142887
8       v9    0.144182
10     v11    0.184374
9      v10    0.356971

[121 rows x 2 columns]
Accuracy: 0.9125
Precision: 0.9127423918294114
Recall: 0.9125
F1-score: 0.9125005031251259
```

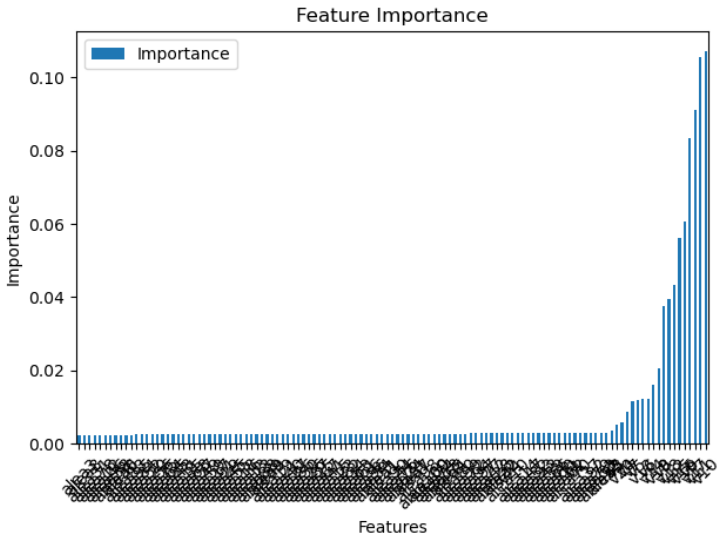
四、結論

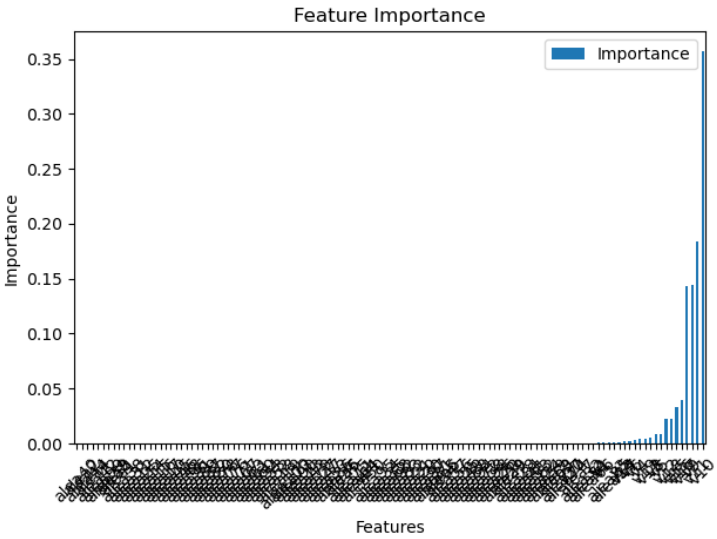
在進行特徵重要性分析時，發現用 Logistic Regression Model、lasso 方法及 Linear SVM Model、ridge 迴歸方法，做出來的結果分佈雷同，預測變量的重要性較噪音變量來得大，屬負項類別的變數較多，模型預測的結果傾向於負向類別



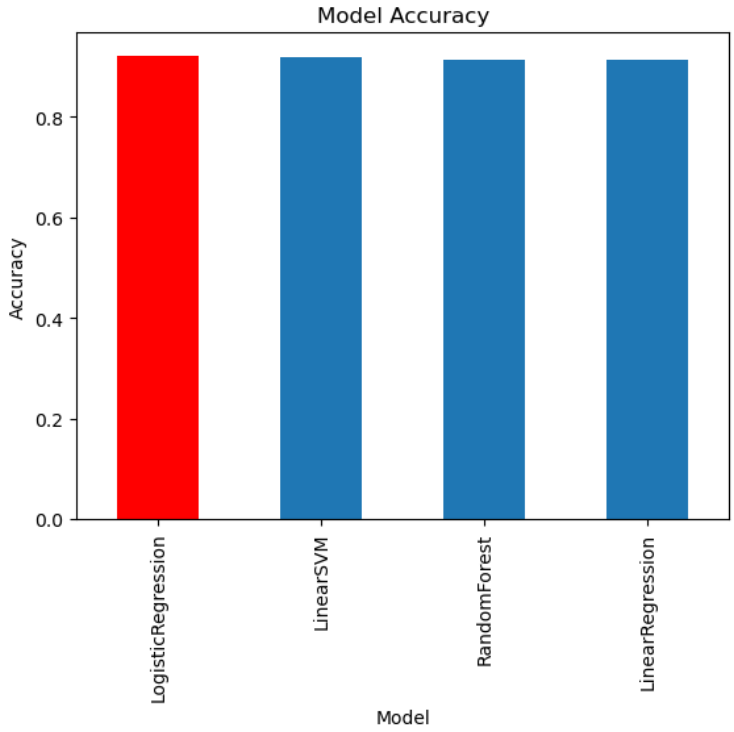


用 RandomForest Model、Linear Regression Model、Grafting technique 方法，做出來的結果分佈雷同，預測變量的重要性較噪音變量來得大，屬正項類別的變數佔大多數，模型預測的結果傾向於正向類別





在比較模型對於測試集中的樣本正確預測的比例時，各個 Model 表現的準確度都相當不錯，至少都有 90% ，最好的為 Logistic Regression Model，來到 92.15%，其餘指標的排序也都為 LogisticRegression > LinearSVM > RandomForest > LinearRegression


Model Accuracy
Accuracy
0.0
0.2
0.4
0.6
0.8
LogisticRegression
LinearSVM
RandomForest
LinearRegression
Model