SCC403 Data Mining Coursework

36228559

MSc Data Science

***Abstract*— In this study, four common methods of analysis are explored in data prospecting, containing two types of cluster analysis and two types of classification analysis. The cluster analysis part uses K-means for the partitioning method and DBSCAN for the density method to deal with clustering problems. The classification part of the analysis makes use of logistic regression and support vector machines, two well-known algorithms, to deal with classification problems.**

Keywords—data mining, k-means, DBSCAN, logistic regression, SVM

I. INTRODUCTION

In the era of big data, there are many different analytics methods available. There are two main types of analytics, that is cluster analytics and classification analytics. Among the cluster analysis algorithms, the K-means algorithm is the most common clustering algorithm. It is often used as a basis for evaluation in market segmentation, machine vision, geostatistics, astronomy and agriculture. However, Ikotun et al., (2023) mention that the K-means algorithm sometimes has a negative impact on performance. This is because the algorithm requires the user to specify the number of clusters in the initial process and to choose the initial cluster centers at random. This makes the performance of the algorithm vulnerable to the selection of this initial cluster.

Therefore, this article uses another cluster analysis method, DBSCAN, a density-based clustering non-parametric algorithm, its main concepts are given a set of data points in a characteristic space, the algorithm divides the nearby points into groups (points with many neighbors) and marks the outlier points in the low-density region (i.e. the closest points are also very far away), so that points farther away from the aggregation are removed as noise (Majhi & Biswal, 2018).

As for classification analytics, two common classification algorithms, logistic regression and *support vector machines*(SVM), are used. In terms of the objective function, the difference is that logistic regression uses logistical loss and SVM uses hinge loss (Xu et al., 2017). Both of which aim to increase the weight of data points that are more relevant to the classification and decrease the weight of data points that are less relevant to the classification. The SVM approach is to learn the classifier by considering only the support vectors, that is, the minority of points most relevant to the classification. In contrast, logistic regression, through non-linear mapping, greatly reduces the weight of points that are far from the classification plane, and relatively increases the weight of data points that are most relevant to the classification.

II. PRE-PROCESSING

In practice, data often suffers from incomplete data, missing data and noise. A dirty data induced by these problems will cause the subsequent model to be unstable. Therefore, after getting the raw data, apart from understanding the underlying attributes of the data, it is also necessary to pre-process the data. The scope of pre-processing includes cleaning and standardising the data in order to maximise the effectiveness of the subsequent analysis.

*A. **Data Cleaning***

Firstly, in relation to dataset 1, no missing values were found and the data distribution for each feature was normal with no obvious outliers. However, as the data has 18 dimensions and there is significant multicollinearity, Principal Component Analysis (PCA) was chosen to reduce the dimensionality of the data, which not only presents the data better but also saves a lot of time consumed by the model.

For dataset 2, the first 16 data items only contained data from object 1, so there was no way to compare them with object 2, so they were removed. Whereas in the dataset2, the three features presented a high degree of linear correlation with each other and there was also the issue of collinearity, so each feature was normalised and PCA was used.

*B. **Principal Component Analysis (PCA)***

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique. It works by first projecting the data along the direction of maximum variance and then iterating so that the data is compressed to the largest single dimensional vector, after which the explained variance of each level of compression is examined and the maximum level of interpretation is selected to determine the final dimensionality of PCA used in the posterior model. Refer to Cai (2017) to maximize the variance between the data after projection as a perspective. Assume that the matrix $X \epsilon R^{d\times m}$ is the original matrix, the matrix $Z \epsilon R^{d\times e}$ is the matrix obtained after projection, and the matrix $W \epsilon R^{d\times d}$ is the projected matrix.

The equations regarding PCA are mainly the variance between the projected points is the distance between the points and the origin, i.e. the sum of the absolute values of the inner products of the original and projected vectors. If let w_1 be the direction of the first axis of the projection, the objective function of the PCA can be written as:

$$max \frac{1}{m} \sum_{i=1}^{m} |\overrightarrow{x_i} \cdot \overrightarrow{w_1}|$$

$$s.t.\ \ \overrightarrow{w_1} \cdot \overrightarrow{w_1} = 1$$

Returning to the results of the dataset1 execution, the distribution of each feature was normalised after performing the normalisation. As a rule of thumb, setting the criterion to PCA must have at least 80% of the interpretable variance. Therefore, as shown in Figure 1, it was chosen to reduce the dimensionality to N=3 to achieve the pre-determined condition.

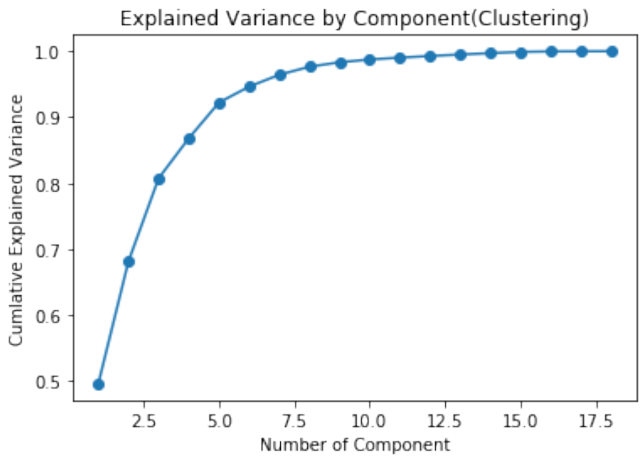

Fig. 1. Explained Variance by Component(Clustering)

The PCA dimension in dataset 2 was determined by explaining the variance, as shown in figure2. In processing PCA of dataset 2 is downscaled to N=2 at 95% of the explainable variance level.

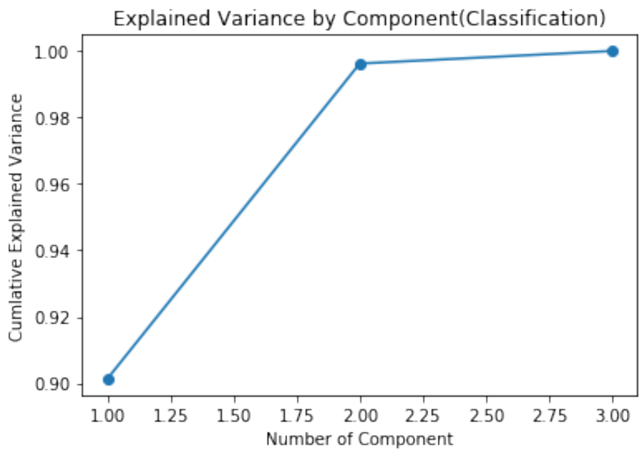

Fig. 2. Explained Variance by Component(Classification)

III. CLUSTERING

Cluster analysis is done by static classification, where similar data is divided into more subsets to make the data interpretable. In more detail, Based on Majhi & Biswal (2018), there are four main types of cluster algorithms, namely partitioning methods, hierarchical methods, grid based methods and density based methods. The essay focuses on the K-meams which from partitioning methods and DBSCAN which is attributed to density based methods to deal with unsupervised learning problems. This essay also draws on Zhou et al., (2014) seven-stage implementation framework for dealing with cluster analysis methods.

A. K-means

K-means clustering, in simple terms, is a clustering method that separates the data set being analyzed into K separate sub-clusters and separates the data by the degree of difference (Mikołajewski et al., 2022).

For reference to Ay et al. (2023), the algorithmic flow of k-means is roughly as follows. Firstly, it should be set into K clusters, and then K cluster centers are randomly set in the feature space. Secondly, the Euclidean Metric of each data point to each of the K cluster centers is calculated. Then the data is assigned to the nearest cluster center after the PCA dimensionality reduction process. Next, after all the data points have been assigned, each group is updated by averaging the data points that have just been assigned. Finally, this is repeated until the end of the cluster.

This essay uses the elbow method to determine the size of K. First, WCSS decreases as the number of clusters increases. Within Cluster Sum of Squares (WCSS) is the sum of the squares of the distances from each point to the center of the cluster, and then the knee point is found, i.e. the place where the bend is the strongest. The position of this knee point determines the number of clusters (K value). The final value of k is set to 4 according to Figures 3. Figures 4 and 5 present the training results of the k-means model in two and three dimensions respectively.

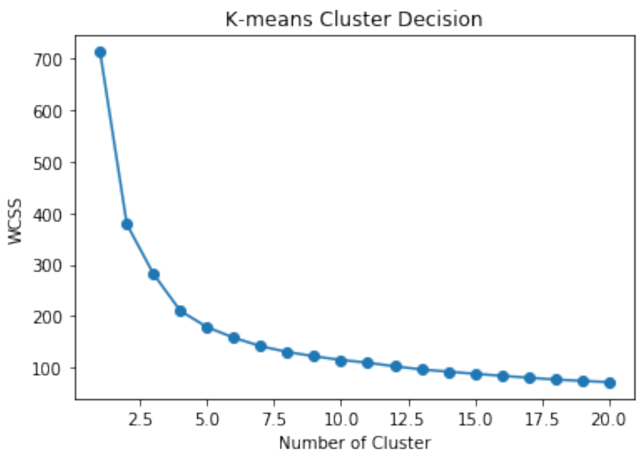

Fig. 3. K-mean Knee Point

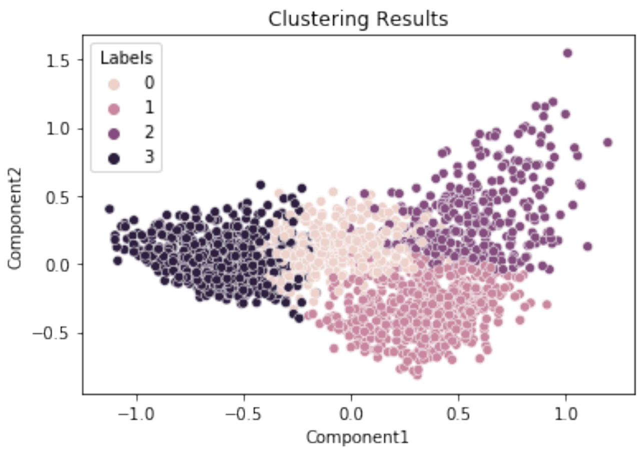

Fig. 4. K-mean Clustering(2D)

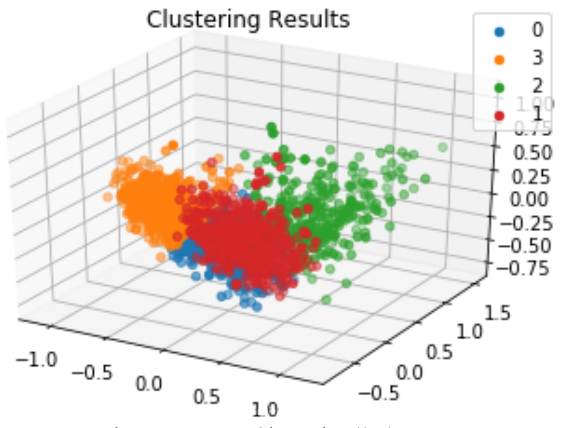

Fig. 5. K-mean Clustering(3D)

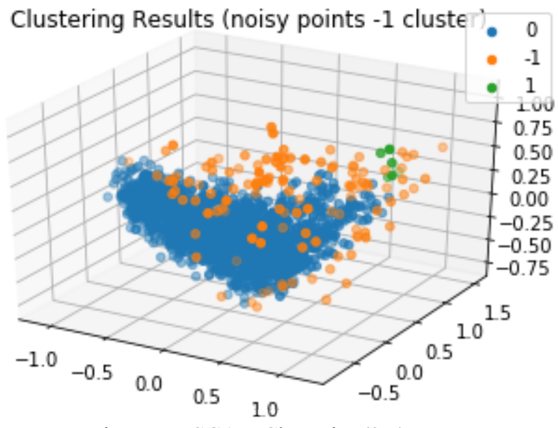

Fig. 7. DBSCAN Clustering(3D)

B. *Density-based spatial clustering of applications with noise (DBSCAN)*

DBSCAN is a density-based spatial clustering for noisy applications, which does not require a specified number of clusters and being able to find outliers as well as can cluster dense datasets of any shape. This advantage is superior to the performance of K-means in detecting irregularly shaped clusters without prior knowledge of the number of clusters (Wu et al., 2022).

According to the description Wu et al. (2022) provided, DBSCAN executes the procedure by starting from A point on its own, then searching for data within the eps range around A. If there are more than min_samples of data in the current eps range, we will consider A to be a Core and then start doing the same for other data within A's eps range until there are no more min_samples in the eps range of a point and then we will stop. If the start does not find enough points around the very beginning, it is judged as Noise. Figures 6 and 7 present the training results of the DBSCAN model in two and three dimensions respectively.

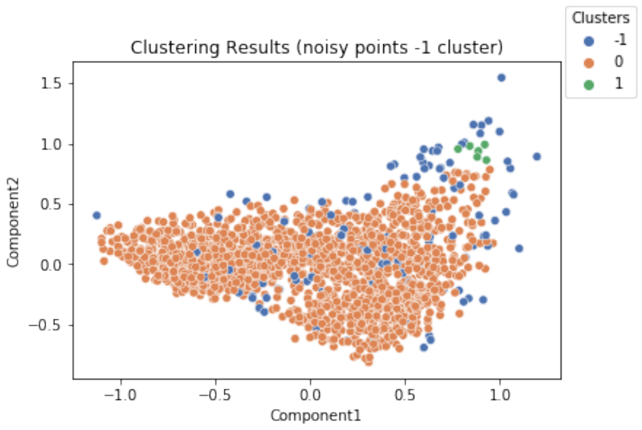

Fig. 6. DBSCAN Clustering(2D)

IV. Classification

The aim of the supervised classification algorithm is to assign a class label to each input example. The goal is to find a learned model that can be trained and used for new, unseen examples.

A. *Logistic Regression (LR)*

The logistic regression equation converts the class target variables into logit odds values for the event to predict the linear relationship between the dependent and predicted variables. The logit odds of the event are then converted to between 0 and 1 by the sigmoid function. By reference to Jaya Hidayat et al. (2022) provided the Logistic Regression can be defined as the equation below:

$$P(x) = \frac{exp(w \cdot x + b)}{1 + exp(w \cdot x + b)}$$

where w is the coefficient matrix, x is the feature matrix and b is the intercept, and $w \cdot x$ is the dot product from the matrix.

Figure 8 shows the training results for the Logisticus regression model, with the decision boundaries showing that only one data has been miscategorised.

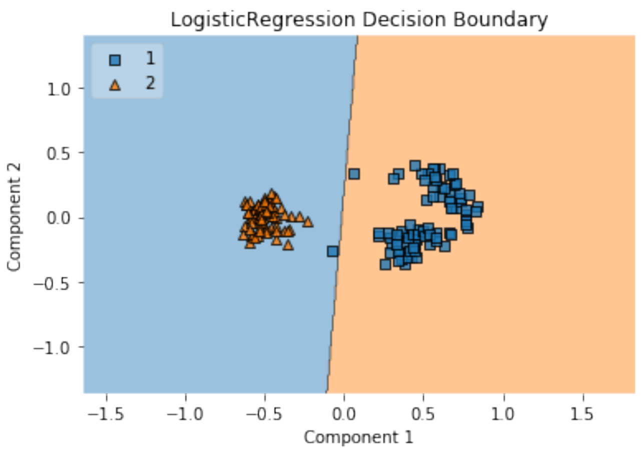

Fig. 8. LR Decision Boundary

In order to evaluate the model more effectively, Figure 9 further uses a confusion matrix to quantitatively evaluate the performance of the model.

Figure 9 shows the confusion matrix with TP on the top left, FN on the top right, FP on the bottom left, and TN on the bottom right, the meanings of which are presented below.

TP - True Positive: the actual category is 0, and the judgement is 0 (correct)
FN - False Negative: the actual category is 0, but the judgement is 1 (incorrect)
TN - True Negative: Actual is 1 and the judgement is 1 (correct)
FP - False Positive: actually 1, but judged to be 0 (incorrect)

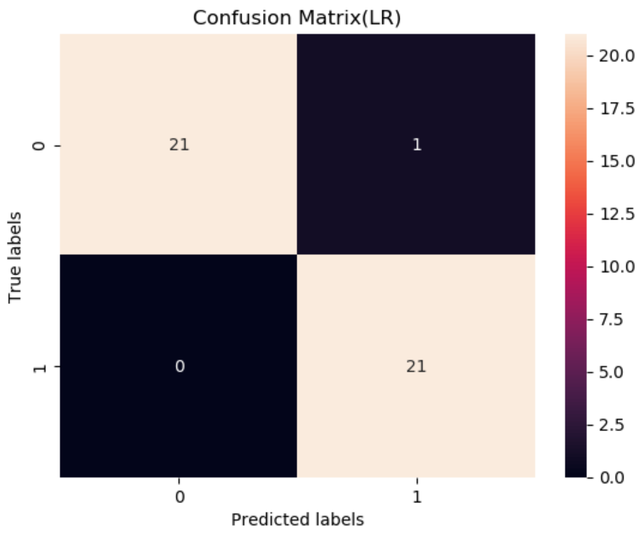

Fig. 9. LR Confusion Matrix

B. *Support Vector Machine (SVM)*

SVM is a supervised learning method that uses the principle of statistical risk minimization to estimate a categorical hyperplane, based on the concept of finding a decision boundary that maximizes the margins between two classes so that they can be perfectly separated. Citing Ertekin et al. (2011), the formula states that the equation for the SVM algorithm is as follows:

$$min\ \mathrm{F}(w,b) = \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{n} \varepsilon_i$$
$$with\ \forall_i \{y_i(w \cdot \phi(x_i) + b) \geq\ 1 - \varepsilon_i\ , \varepsilon_i \geq 0$$

where w is the normal vector of the projection plane, $\phi(\cdot)$ is the mapping from the original space to the projection space, and ε_i is the parameter that determines the classification criteria.

Figure 10 shows the training results for the SVM model, with the decision boundaries showing that only one data has been miscategorised.

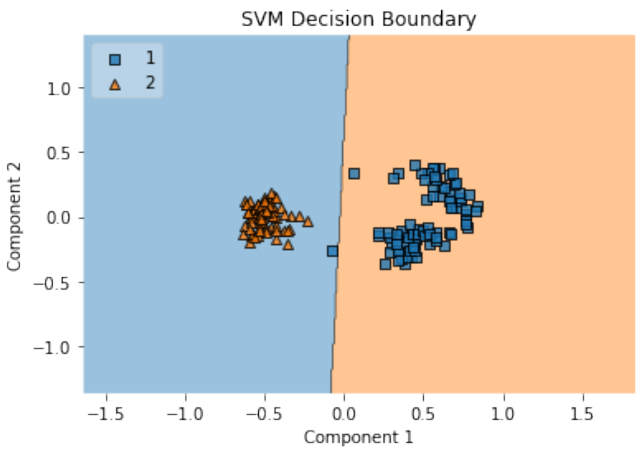

Fig. 10. SVM Decision Boundary

Figure 11 further uses a confusion matrix to quantitatively evaluate the performance of the model.

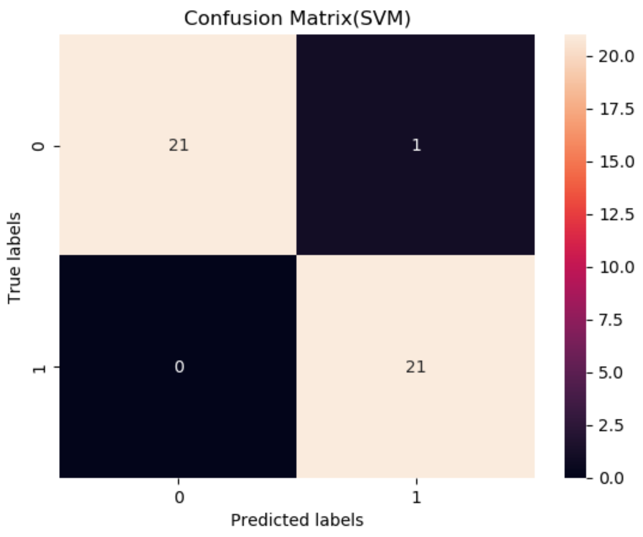

Fig. 11. SVM Confusion Matrix

V. CONCLUSION

To conclude, this essay compares K-means, a partitioning approach, and DBSCAN, a density-based approach, to unsupervised problems. Then apply logistic regression and SVM to supervised learning, the difference being that the former uses logistical loss and SVM uses hinge loss.

For clustering, find that K-means is much more effective than DBSCAN because the data distribution is more concentrated after PCA, and since DBSCAN is based on the concept of density, it tends to treat points with concentrated density as a class, therefore, this information and performance is not good. Two of the four types of clustering algorithms are presented in this essay, and two others are not yet mentioned: hierarchical methods, grid-based methods, and in the future the performance of the four types of clustering algorithms can be further compared based on the dataset used in this paper.

In the classification section, find that Logistic Regression and SVM perform very similarly for this dataset, as the data distribution of the dataset used in this essay is very suitable for linear classifiers. In terms of results, both have the same Accuracy, Precision, Recall and F1 score, and there is no significant difference in model processing time(see Table I for details). The dataset used in this essay is relatively simple, and the differences between the models will be more apparent in the future when they can be extended to higher dimensional data.

TABLE I. presents the performance of this classification, using five indicators: Accuracy, Precision, Recall, F1 Score and Time (sec), which are calculated according to Figure9 and Figure11.

Accuracy = (TP+TN)/(TP+TN+FP+FN), representing the percentage of correct predictions.

Precision = TP/(TP+FP), representing the proportion of samples identified as 0 by the model to the number of 0.

Recall = TP/(TP+FN), representing the proportion of samples correctly identified as 0 by the model to the number of 0.

F1 score = (2×Precision×Recall)/(Precision+Recall), a good algorithm that ideally balances Recall and Precision and tries to have both metrics as high as possible. So there is a set of judgement that can take into account both Recall and Precision.

Time(sec), which represents the length of time the model takes to train.

TABLE I.

	Model performance assessment				
	Accuracy	*Precision*	*Recall*	*F1 Score*	*Time(sec)*
LR	0.98	1.00	0.95	0.97	0.461
SVM	0.98	1.00	0.95	0.97	0.319

References

[1] Ay, M. et al. (2023) “FC-Kmeans: Fixed-centered K-means algorithm,” Expert Systems with Applications, 211, p. 118656.

[2] Cai, W. (2017) “A dimension reduction algorithm preserving both global and local clustering structure,” Knowledge-Based Systems, 118, pp. 191–203.

[3] Ertekin, Ş., Bottou, L. and Giles, C.L. (2011) “Nonconvex online support Vector Machines,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(2), pp. 368–381.

[4] Gao, J., Tao, X. and Cai, S. (2023) “Towards more efficient local search algorithms for constrained clustering,” Information Sciences, 621, pp. 287–307.

[5] Genin, Y.V. (1996) “Euclid algorithm, orthogonal polynomials, and generalized Routh-Hurwitz algorithm,” Linear Algebra and its Applications, 246, pp. 131–158.

[6] Ikotun, A.M. et al. (2023) “K-means Clustering Algorithms: A comprehensive review, variants analysis, and advances in the era of Big Data,” Information Sciences, 622, pp. 178–210.

[7] Jaya Hidayat, T.H. *et al.* (2022) “Sentiment analysis of Twitter data related to Rinca Island development using doc2vec and SVM and logistic regression as classifier,” *Procedia Computer Science*, 197, pp. 660–667.

[8] Majhi, S.K. and Biswal, S. (2018) “Optimal cluster analysis using hybrid K-means and Ant Lion optimizer,” Karbala International Journal of Modern Science, 4(4), pp. 347–360.

[9] Mikołajewski, K. et al. (2022) “Development of Cluster Analysis Methodology for identification of model rainfall hyetographs and its application at an urban precipitation field scale,” Science of The Total Environment, 829, p. 154588.

[10] Wu, G. et al. (2022) “HY-DBSCAN: A hybrid parallel DBSCAN clustering algorithm scalable on distributed-memory computers,” Journal of Parallel and Distributed Computing, 168, pp. 57–69.

[11] Xu, G. et al. (2017) “Robust support vector machines based on the rescaled hinge loss function,” Pattern Recognition, 63, pp. 139–148.

[12] Zhou, Y. et al. (2014) “A cluster-based method to map urban area from DMSP/OLS nightlights,” Remote Sensing of Environment, 147, pp. 173–185.

APPENDIX A

Key diagrams of the research process

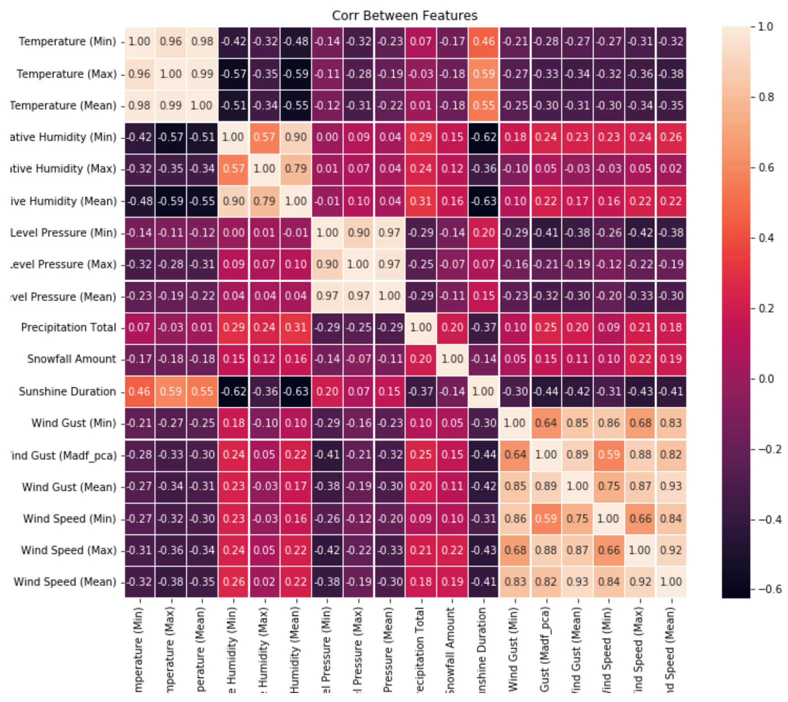

Figure A1. Correlation between features in the cluster data set

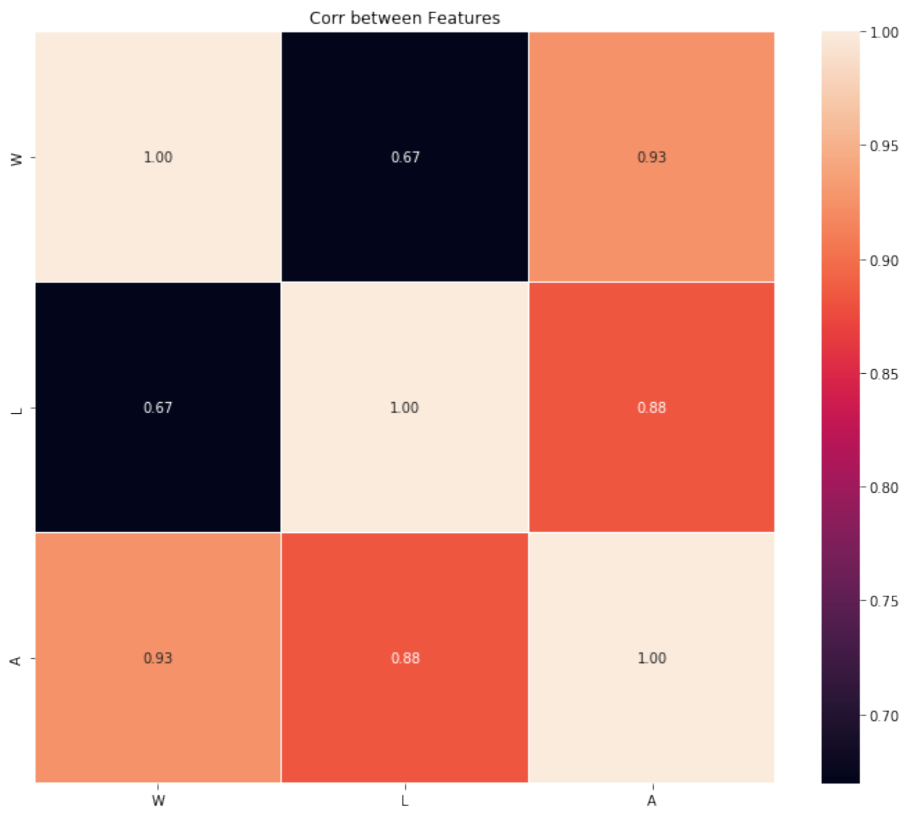

Figure A2. Correlation between features in the classification data set

APPENDIX B

Clustered Data Code

```
# # SCC403 Dataset 1
# ### 1. Import library & data
# In[1]:
get_ipython().run_cell_magic('capture', '', 'import pandas as pd\nimport numpy as np  \nimport matplotlib.pyplot as plt\nimport seaborn as sns\nfrom sklearn import preprocessing   \nfrom sklearn.cluster import KMeans\nfrom sklearn.decomposition import PCA\nfrom sklearn.cluster import DBSCAN')
# In[2]:
colnames = ['Temperature (Min)','Temperature (Max)','Temperature (Mean)','Relative Humidity (Min)',
'Relative Humidity (Max)','Relative Humidity (Mean)','Sea Level Pressure (Min)','Sea Level Pressure (Max)',
'Sea Level Pressure (Mean)','Precipitation Total','Snowfall Amount','Sunshine Duration','Wind Gust (Min)',
'Wind Gust (Madf_pca)','Wind Gust (Mean)','Wind Speed (Min)','Wind Speed (Max)','Wind Speed (Mean)']
df = pd.read_csv('Data Files/ClimateDataBasel.csv', names=colnames, header=None)
# ### 2. Data pre-processing
# After checking the data, there are no missing values and no obvious outlier
# In[3]:
df.info()
df.describe().T
# Several of these features have collinearity
# In[4]:
fig, ax = plt.subplots(figsize=(12,10))
sns.heatmap(df.corr(), annot=True, fmt=".2f", linewidths=0.3)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.title("Corr Between Features")
#plt.savefig("Plots/corr.png")
# Normalization
# In[5]:
df_scale = preprocessing.MinMaxScaler().fit_transform(df)
df_scale.shape
# ### 3.Dimensionality reduction with PCA
# To get more than 80% of variance explained I need 3 principal components.
# In[6]:
pca = PCA()
pca.fit(df_scale)
plt.plot(range(1,19), pca.explained_variance_ratio_.cumsum(), marker='o')
plt.title("Explained Variance by Component(Clustering)")
plt.xlabel("Number of Component")
plt.ylabel("Cumlative Explained Variance")
# In[7]:
pca = PCA(n_components=3)
pca.fit(df_scale)
df_pca = pca.transform(df_scale)
df_pca = pd.DataFrame(df_pca,columns=['Component1','Component2','Component3'])
# In[8]:
plt.figure(figsize=(10,10))
var = np.round(pca.explained_variance_ratio_*100, decimals = 1)
plt.bar(x=range(1,len(var)+1), height = var, tick_label = df_pca.columns)
plt.title("Explained Variance by Components")
plt.ylabel("Explained Variance(%)")
plt.show()
# In[9]:
from mpl_toolkits.mplot3d import axes3d, Axes3D #<-- Note the capitalization!
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(projection='3d')
ax.scatter(df_pca.iloc[:, 0], df_pca.iloc[:, 1], df_pca.iloc[:, 2])
plt.title("Data Distribution")
ax.set_xlabel('Component1')
ax.set_ylabel('Component2')
ax.set_zlabel('Component3')
plt.show()
# ### 4. Model training
# K-means
# In[10]:
get_ipython().run_cell_magic('capture', '', "wcss = []\nfor i in range(1,21):\n    kmeans_pca = KMeans(n_clusters=i, init='k-means++',random_state=42)\n    kmeans_pca.fit(df_pca)\n    wcss.append(kmeans_pca.inertia_)")
# In[11]:
plt.plot(range(1,21),wcss, marker='o')
plt.title("K-means Cluster Decision")
plt.ylabel("WCSS")
plt.xlabel("Number of Cluster")
# The kink comes at the 4 clusters mark. So, we'll be keeping a four-cluster solution.
# In[12]:
get_ipython().run_cell_magic('capture', '', "kmeans_pca = KMeans(n_clusters=4, init='k-means++', random_state=42)\nkmeans_pca.fit(df_pca)")
# In[13]:
df_pk = pd.concat([df.reset_index(drop=True),pd.DataFrame(df_pca)],axis=1)
df_pk.columns.values[-3:] = ['Component1','Component2','Component3']
df_pk['Labels'] = kmeans_pca.labels_
df_pk.head()
# In[14]:
sns.scatterplot(data=df_pk, x='Component1',y='Component2',hue='Labels')
plt.title("Clustering Results")
plt.show()
fig = plt.figure()
```

```
ax = fig.add_subplot(111, projection='3d')
plt.title("Clustering Results")
for s in df_pk.Labels.unique():

ax.scatter(df_pk.Component1[df_pk.Labels==s],df_pk.Co
mponent2[df_pk.Labels==s],df_pk.Component3[df_pk.La
bels==s],label=s)
ax.legend()
# DBScan
# In[15]:
from sklearn.neighbors import NearestNeighbors
nearest_neighbors = NearestNeighbors(n_neighbors=6)
neighbors = nearest_neighbors.fit(df_pca)
distances, indices = neighbors.kneighbors(df_pca)
distances = np.sort(distances[:,5], axis=0)
# In[16]:
from kneed import KneeLocator
i = np.arange(len(distances))
knee = KneeLocator(i, distances, S=1, curve='convex',
direction='increasing', interp_method='polynomial')
fig = plt.figure(figsize=(5, 5))
knee.plot_knee()
plt.xlabel("Points")
plt.ylabel("Distance")
print(distances[knee.knee])
# The knee occurs at approximately 0.14
# For multidimensional dataset, minPts should be 2 *
number of dimensions
# In[17]:
clusters = DBSCAN(eps=0.14,
min_samples=6).fit(df_pca)
p = sns.scatterplot(data=df_pca, x=df_pca.iloc[:,0],
y=df_pca.iloc[:,1], hue=clusters.labels_, legend="full",
palette="deep")
sns.move_legend(p, "upper right", bbox_to_anchor=(1.17,
1.2), title='Clusters')
plt.title("Clustering Results (noisy points -1 cluster)")
plt.show()
df_test =
pd.concat([df_pca.reset_index(drop=True),pd.DataFrame(
clusters.labels_,columns=['Labels'])],axis=1)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for s in df_test.Labels.unique():

ax.scatter(df_test.Component1[df_test.Labels==s],df_test.
Component2[df_test.Labels==s],df_test.Component3[df_t
est.Labels==s],label=s)
plt.title("Clustering Results (noisy points -1 cluster)")
ax.legend()
```

Classification Data Code

```
# # SCC403 Dataset 2
# ### 1. Import library & data
# In[1]:
get_ipython().run_cell_magic('capture', '', "import pandas
as pd\nfrom sklearn import preprocessing\nfrom
sklearn.decomposition import PCA\nimport
matplotlib.pyplot as plt\nimport time\nfrom
sklearn.neural_network import MLPClassifier\nfrom
sklearn import datasets\nfrom sklearn.tree import
DecisionTreeClassifier\nfrom sklearn.model_selection
import KFold, cross_val_score\nfrom
sklearn.linear_model import LogisticRegression\nfrom
sklearn.model_selection import train_test_split\nfrom
sklearn.model_selection import cross_val_predict\nfrom
sklearn.metrics import confusion_matrix\nimport seaborn
as sns\nfrom sklearn.svm import SVC\nfrom
sklearn.model_selection import ShuffleSplit\nfrom
sklearn.metrics import classification_report\nfrom
mlxtend.plotting import
plot_decision_regions\nplt.rcParams.update(plt.rcParams
Default)\n\n\nWLA_col = ['W','L','A']\nLabels_col =
['Labels']\nX = pd.read_csv('Data Files/WLA.csv',
names=WLA_col, header=None)\ny = pd.read_csv('Data
Files/Labels.csv', names=Labels_col, header=None)")
# ### 2. Data pre-processing
# The first 16 data only motorcycle data, can not be
compared with the car, so delete
# In[2]:
X = X.iloc[16:]
y = y.iloc[16:]
# Several of these features have collinearity
# In[3]:
fig, ax = plt.subplots(figsize=(12,10))
sns.heatmap(X.corr(), annot=True, fmt=".2f",
linewidths=0.3)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.title("Corr between Features")
# Normalization
# In[4]:
X_scale =
preprocessing.MinMaxScaler().fit_transform(X)
X_scale.shape
# ### 3. Dimensionality reduction with PCA
# To get more than 90% of variance explained I need 2
principal components.
# In[5]:
pca = PCA()
pca.fit(X_scale)
plt.plot(range(1,4),
pca.explained_variance_ratio_.cumsum(), marker='o')
plt.title("Explained Variance by
Component(Classification)")
plt.xlabel("Number of Component")
plt.ylabel("Cumlative Explained Variance")
# In[6]:
pca = PCA(n_components=2)
pca.fit(X_scale)
X_pca = pca.transform(X_scale)
X_pca =
pd.DataFrame(X_pca,columns=['Component1','Compone
nt2'])
# In[7]:
sns.scatterplot(data=X_pca,x=X_pca.iloc[:,0],y=X_pca.ilo
c[:,1])
plt.title("Data Distribution")
plt.show()
# ### 4. Model training
# Split the data
# In[8]:
```

```
X_train, X_test, y_train, y_test = train_test_split(X_pca,
y, random_state=0)
# LogisticRegression and cross validation
# In[9]:
start = time.time()
LR = LogisticRegression()
LR.fit(X_train, y_train)
y_pred = LR.predict(X_test)
cv = ShuffleSplit(n_splits=10, test_size=0.2)
scores = cross_val_score(LR, X, y, cv=10)
cm = confusion_matrix(y_test, y_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix(LR)')
print("LogisticRegression accuracy after 10 fold CV:
%0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print("Training time: "+ str(round(time.time() - start, 3)) +
"s")
# In[10]:
plot_decision_regions(X_pca.to_numpy(),
y.Labels.to_numpy(), clf=LR, legend=2)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('LogisticRegression Decision Boundary')
plt.show()
# SVM and cross validation
# In[11]:
start = time.time()
SVM = SVC(kernel='linear')
SVM.fit(X_train, y_train)
y_pred = SVM.predict(X_test)
cv = ShuffleSplit(n_splits=10, test_size=0.2)
scores = cross_val_score(SVM, X, y, cv=10)
cm = confusion_matrix(y_test, y_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix(SVM)')
print("LogisticRegression accuracy after 10 fold CV:
%0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print("Training time: "+ str(round(time.time() - start, 3)) +
"s")
# In[12]:
plot_decision_regions(X_pca.to_numpy(),
y.Labels.to_numpy(), clf=SVM, legend=2)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('SVM Decision Boundary')
plt.show()
```