REPORT ON

ECG_SIGNAL DATASET BY DATA MINING

Y.S. VINAYAKA

521569

INTRODUCTION:

The purpose of this report is to present the results and main strategies of DATA MINING project using ecg_signal Dataset. The goal of the project was to achieve a high accuracy and classifying into different categories. Data Mining techniques help extract meaningful insights from large volumes of data.

DATASET OVERVIEW:

This dataset containing 31 features was obtained by processing the provided ECG signals all are numerical, and the target variable is also encoded. we load the dataset which was in CSV format with Pandas dataframe.

IMPORTING LIBRARIES:

To accomplish the DATA MINING task on the ECG_SIGNAL Dataset, several libraries were utilized. These libraries provided essential functionalities for data handling, model creation, training, and evaluation. The following libraries were imported:

DATFRAME: ECG_SIGNAL

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns'
from matplotlib.pyplot import figure
from sklearn.neighbors import KNeighborsClassifier
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SequentialFeatureSelector
import sklearn
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.model_selection import GridSearchCV
from sklearn.tree import export_text
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
```

These libraries played a crucial role in various stages of the project, facilitating data handling, model creation, and result analysis. The functionalities provided by these libraries, the project workflow was streamlined and efficient, enabling smooth implementation of the data mining task.

PRE-PROCESSING:

After loading the dataset, we have created the data to new dataframe. Using basics of pandas, we displayed pandas properties like (info(), shape, describe(), isnull() etc) and we converted all columns of the dataset to float.

To check the duplicate of the data, we have used the interquartile range to detect outliers (IQR). The data points used to fall out of IQR 0.25 and 0.75 are considered as outliers. We have performed the outlier's detection for each class data and replaced the missing value with the median. We obtained data is plotted with Boxplot which creates box and whisker plot for each column of the dataframe. We visualized the data using histogram plot for each column of the data.

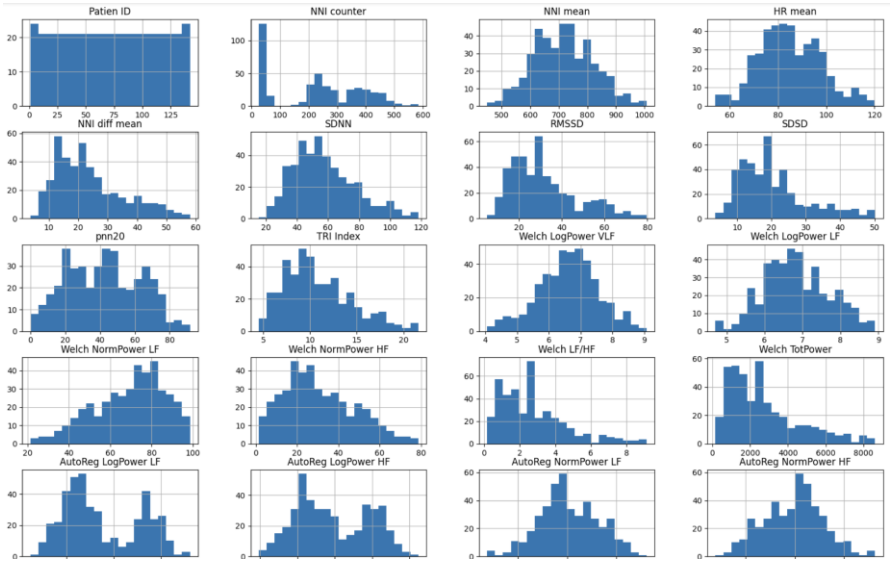
```
[13] def detect_outliers(df, feature_name):
       Q1 = df[feature_name].quantile(0.25)
       Q3 = df[feature_name].quantile(0.75)

       IQR = Q3- Q1

       lwr_bound = Q1-1.5 * IQR
       upp_bound = Q3+1.5*   IQR

       ls = df.index[np.logical_or(df[feature_name]<lwr_bound, df[feature_name]>upp_bound)]

       return ls
```

```
[14]
     outliers_detected = {}
     for i in ecg_df.columns:
       outliers = detect_outliers(ecg_df, i)
       outliers_detected[i] = outliers

       print('variable', i)
       print(outliers)
       print(ecg_df[i].iloc[outliers])
       print('\n')
```

```
variable Patien ID
Int64Index([], dtype='int64')
Series([], Name: Patien ID, dtype: float64)


variable NNI counter
Int64Index([], dtype='int64')
Series([], Name: NNI counter, dtype: float64)
```

DATA VISUALIZATION:

We have visualized the data to be sure which indexes are really outliers, it is convenient to plot the histogram.

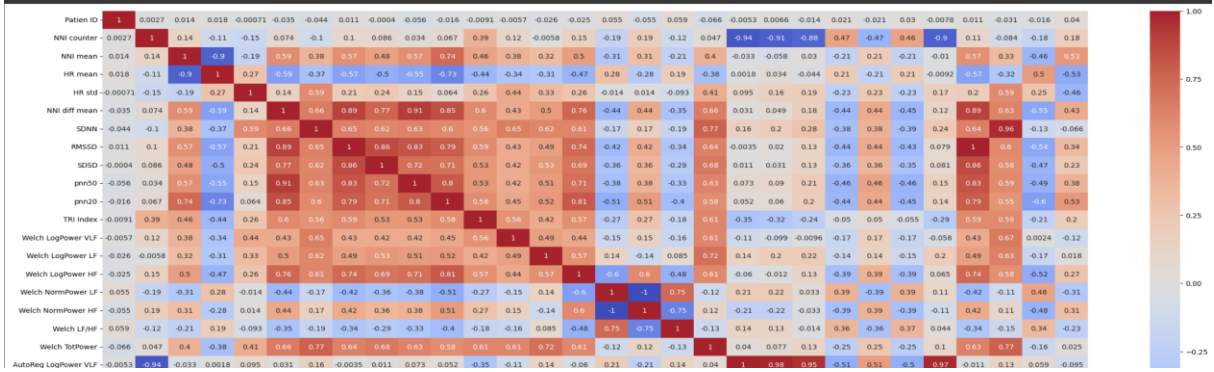


CORRELATION:

We have separated the dataframe with all the columns into X variables and the column ‘label’ to another variable Y which is last column of the dataframe and it also contains the target variable of the output variable of the dataset. Typically, this kind of split is done to separate the input variable from the output variable so that we can use input variables to predict the output variable. We used the Heatmap to check the correlation between features.

And noticing the data has highly negative correlated values with the target, we have extracted the features that are positively linearly correlated with the target variable. The threshold value taken is 0.7 as x and plotted the heatmap of the correlation.

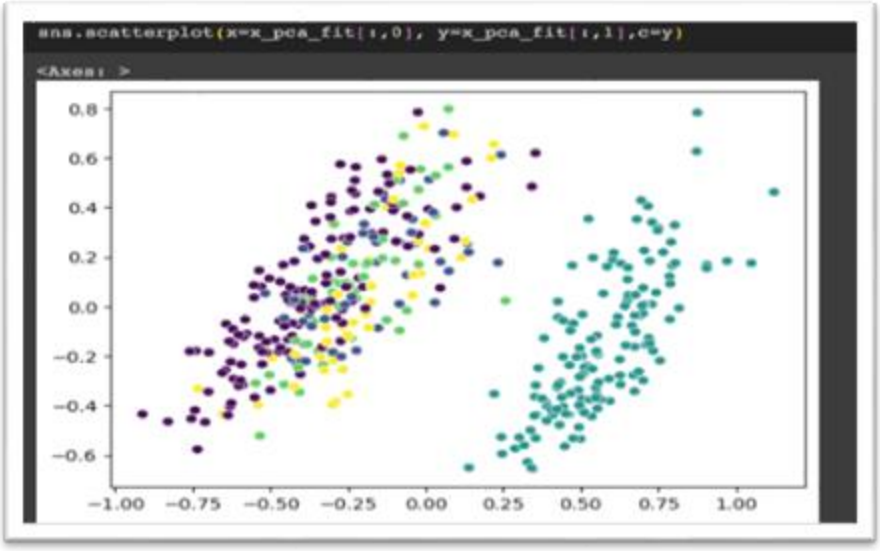
```
corrmat = X.corr()
plt.figure(figsize=(30,15))
sns.heatmap(corrmat,annot=True,cmap="coolwarm")
plt.show()
```

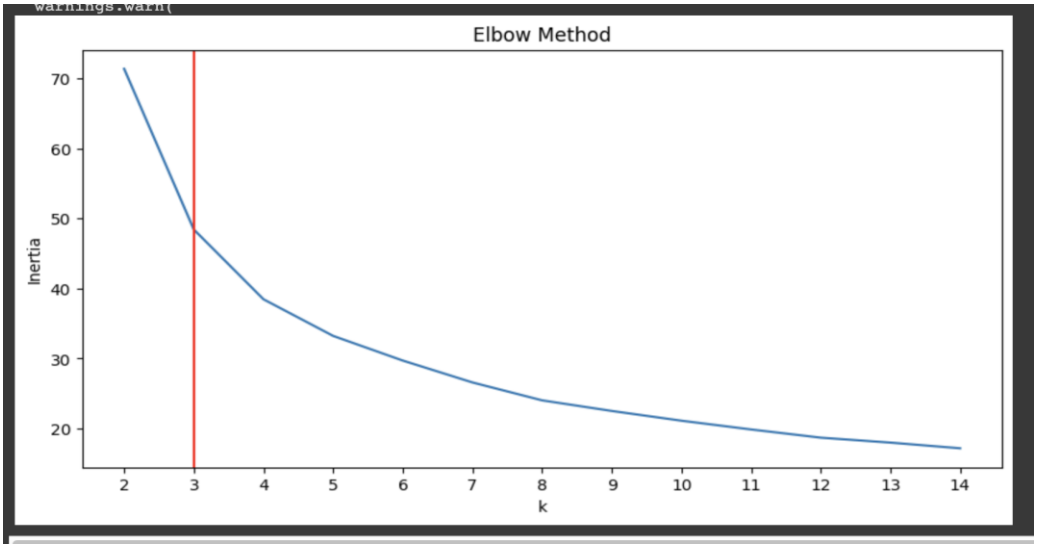


FEATURE SELECTION AND EXTRACTION:

For Feature selection, we have used multiple feature selection techniques such as SelectKbest, SelectFromModel, Sequential Feature selector and also SelectFromModel embedded method to find the most important features that are more important to the target variable. The features that were more common in all the 5 feature selection models done have been selected.

If in any case where two or more independent features have the same correlation with the target variable and have equal correlation with each other, the feature selection model fails to select the best feature. In such case, feature exraction is used to extract the best feature from them. PCA is used to reduce the number of features within the dataset while still retaining as much information as possible. To determine the number of dimensions the PCA should reduce the data, an iteration method was used, which starts by considering one component and increases its components after every iteration.





The number of clusters are selected at the point where the elbow is created.

TRAINING AND TESTING THE MODEL:

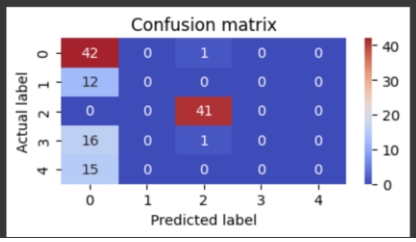
The data is then split into Train and Test data with 70% and 30% respectively using Traintestsplit with Scikit learn. Various Machine Learning models are used to check which model has the best performance. The classification models used for detecting the emotions are: Logistic Regression,KNeighboursClassifier,SVM Classifier and DecisionTreeClassifier.

We compared the each models, we got accuray for each model. Logistic Regression is 75%, KNeighbour Classifier is 81%, SVM Classifier is 79%, and Decision Tree Classifier is 69%.

CONFUSION MATRIX:

We also obtained, the selected models are evaluated using Confusion matrix to check whether precision and recall are balanced. The f1 score is used to evaluate the precision and recall balance.

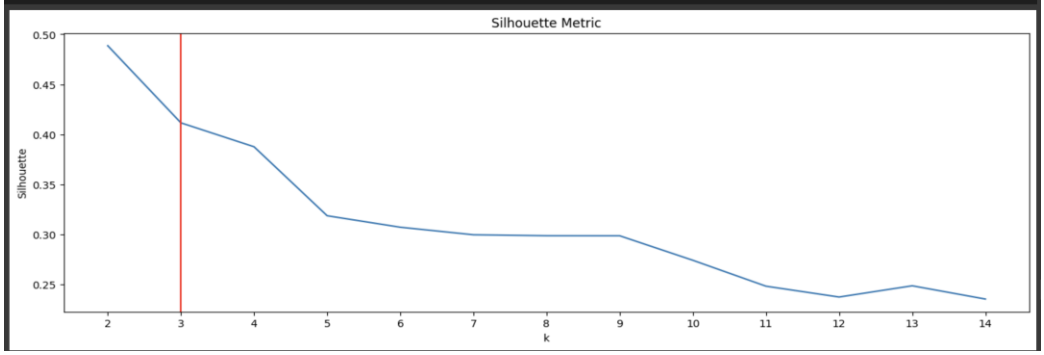
```
[89] cnf_matrix = metrics.confusion_matrix(y_test, lr_pred)
     print(cnf_matrix)

     [[42  0  1  0  0]
      [12  0  0  0  0]
      [ 0  0 41  0  0]
      [16  0  1  0  0]
      [15  0  0  0  0]]

[91] plt.figure(figsize=(4,2))
     sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="coolwarm" ,fmt='g')

     plt.tight_layout()
     plt.title('Confusion matrix', y=1)
     plt.ylabel('Actual label')
     plt.xlabel('Predicted label')
     plt.show()
```



UNSUPERVISED LEARNING:

Unsupervised learning is also used to evaluate the model performance. KMeans clustering and Hierarchical clustering is used as a unsupervised learning techniques. The accuracy is evaluated on the original label data of emotions and predicted classes. The accuracy obtained from KMeans is 75% and from Hierarchical clustering is 34%. Silhouette score is used to evaluate how well the points are clustered inside the cluster, it is calculated between 0 and 1 where 1 being the best score.

```
plt.figure(figsize = (16,5))
plt.plot(silhouette_scores.values())
plt.xticks(range(0,13,1), silhouette_scores.keys())
plt.title("Silhouette Metric")
plt.xlabel("k")
plt.ylabel("Silhouette")
plt.axvline(1, color = "r")
plt.show()
```



CONCLUSION:

In conclusion, the results of this experiment provide valuable insight into the performance of different machine learning models for binary classification problems. Model SVM Classifier was found to be the most accurate and can be considered for further analysis and implementation in the future. We had build the same models using PCA dataset and compared the accuracies provided by the algorithm.The steps involved in the project, including data pre-processing , feature selection, data split, model building and training, model comparison, and unsupervised learning were described in detail.