Data – Hypothyroid

The dataset consists of 30 columns (including the class variable). Out of the 29 features, 22 are categorical and 7 are numerical. The Class variable has 4 categories – negative, compensated_hypothyroid, primary_hypothyroid and secondary_hypothyroid.

Data Cleaning

A lot of the values in the data were '?'s. These had to be cleaned before doing any further analysis. The steps that were performed to do this task were:-

- Replacing '?'s with NaN. This helped in getting an exact count of null values in each column.
- Some columns like T3_measured, TBG_measured had their respective numerical columns also. These numerical columns had NaN values for all 'f' category values in the categorical columns. Hence these numerical columns were dropped.
- There was a variable which indicated pregnant or not. Using this, some NaN values in the 'sex' column were fixed.
- Converted the 'age' column to a numeric type (earlier object type). Then checked for outliers in the column and dropped those rows.
- The 'sex' column had 150 missing values. This made up for only 4% of the total data. Hence these rows were dropped in as there was no other appropriate way of filling in these missing values.
- Dropped the 'TBG_measured' column also as it had only 1 value ('f') throughout the entire column.
- Then the index was reset in order to avoid those missing row indices.

Normalization

- The only numerical column was 'age' which was to be normalized in the range of 0 to 1.
- Imported ***sklearn*** library and from that used the ***preprocessing*** package. Using the ***MinMaxScaler()*** in preprocessing package, normalized the numerical data in the range of 0 to 1.
- Stored this normalized value in a new data-frame **df**.

Label encoding

- Label encoding is performed on the categorical variables because all these categorical variables had only 2 categories. Only the categorical variables were extracted to a new data-frame **enc**.
- From ***sklearn.preprocessing*** imported ***LabelEncoder***.

- After encoding, all the categories were converted to 1's and 0's in their respective columns.
- 'age' & 'Class' variable were added to this data-frame and a copy was made. This copy is the processed data-frame.

Before using logistic regression or neural networks, the data has to be split into train & test data. To do this task, ***train_test_split*** package from ***sklearn.model_selection*** is used. The test size is 25% of the data. The arguments for train_test_split are the features, target variable ('class'), test size (0.25), random_state and stratify.

Since the values in the 'class' variable are imbalanced, stratify is used to take care of that problem.

Logistic Regression

- Imported ***LogisticRegression*** package from ***sklearn.linear_model*** library.
- Imported ***accuracy_score*** package from ***sklearn.metrics*** library.
- Ran the logistic regression model.
- Accuracy was 92.27%. Even though the accuracy is high, it is found that the model can only predict 1 class. Hence, this model is not good.
- Imported ***classification_report*** package from ***sklearn.metrics*** library to print the classification report to print the precision, recall & f1-score.

Neural network

- Imported ***MLPClassifier*** package from ***sklearn.neural_network*** library.
- Ran a neural network model with 2 hidden layers having nodes 100 & 60.
- The accuracy was 92.27%.
- The classification report and confusion matrix were printed.

Experimenting the neural network model with different number of hidden layers and nodes

- **1st case (100,60) –** Accuracy= 92.27%
- **2nd case (100,70) –** Accuracy= 92.49%
  This model showed a little higher accuracy and was also able to predict other classes as well.
- **3rd case (100,60,30) –** Accuracy= 92.38%
  Accuracy decreased slightly.
- **4th case (100,50) –** Accuracy= 92.27%
  Accuracy has decreased again.

SVM – Support Vector Machines

- From sklearn.svm imported SVC
- The arguments for SVC() were kernel='poly', degree=5, gamma='scale'. The gamma value was added to take care of warning. It was found that the model showed highest accuracy at degree=5. Accuracy was 92.38%.

PCA – Principal Component Analysis

- From sklearn.decomposition imported PCA
- This method was used to find out the most relevant features and eliminate the rest.
- Found the explained variance for each column. Took only those columns who were the highest and their variance added up to 0.96.
- Separated into train and test data and then performed logistic regression.
- The accuracy did not seem to change (92.27%).

Random Forest

- From sklearn.ensemble imported RandomForestClassifier
- Created a model with n_estimators=200
- After running the model the accuracy was still 92.27%.

Conclusion

While comparing all the above models, the neural network model with 2 hidden layers having 100 & 70 nodes seems to be the most accurate model with an accuracy of 92.49%. Moreover, this model was able to predict other classes also.