**Machine Learning**

**WAQAR KALEEM KHAN – 01-249191-013**

**ASSIGNMENT No.01**

**Q) Choose any data set, explain its features, type of data and how you would apply machine learning on it.**

**Data Source:**

Data has been chosen from Kaggle featured as Pakistan Intellectual Capital.

<https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset>

The data source is a private dataset owned my multiple people while being led by one Debdatta Chatterjee with latest update on March 12th, 2019 at the time of data extraction for this assignment.

**Data Background:**

The Dataset contains list of anonymous Applicants from displaying their earnings and qualifications and also their living standard as well as their dependents in order to correlate some of the features to conclude whether the applicant is viable for loan or not.

**Data Variables:**

1. Loan\_ID (#) -unique val.(Character/Numeric mix)
2. Gender -variable characters
3. Married - variable characters
4. Dependents - numeric
5. Education - variable characters
6. Self\_Employed - (Yes/No)
7. Applicant Income - numeric
8. Co Applicant Income - numeric
9. Loan Amount - numeric
10. Loan Amount Term -numeric
11. Credit History - binary (1/0)
12. Property Area - variable characters
13. Loan Status -Binary (Y/N)

* The data being used consists of 615 entries.
* Label is addressed as “Loan Status”.

**Q) Apply Decision Tree using two different heuristics as well as implement pruning in any tool using any technique with description.**

**Tool of choice:** Anaconda Navigator – Jupiter Notebook.

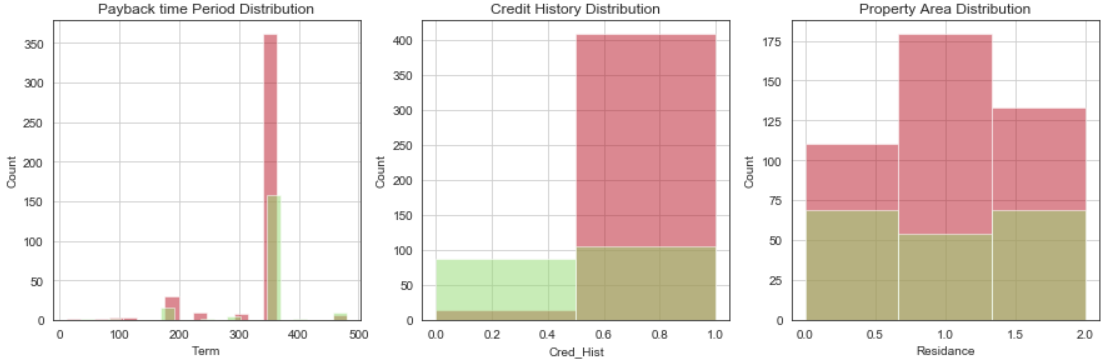
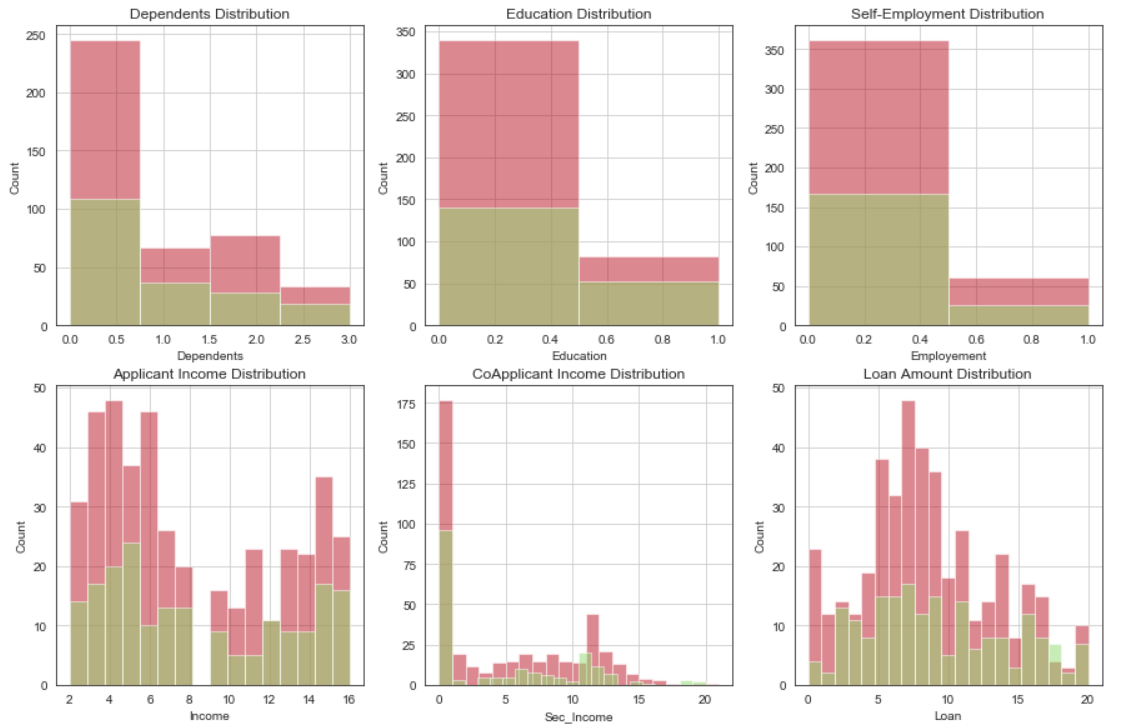
**Language:** Python 3.

**Libraries used:**

* 1. Data Import:
     + - Pandas
  2. Numeric Operations / Preprocessing:
     + - Numpy
       - Sklearn.metrics
       - Sklearn.preprocessing
  3. Plots and Graphs:
     + - Matplotlib
       - Seaborn
       - Graphviz
       - Imageio
  4. Decision Tree:
     + - Sklearn.tree
       - Sklearn.tree.DecisionTreeClassifier
       - Sklearn.model\_selection.train\_test\_split
  5. Saving/Exporting Images:
     + - Io

**Steps followed:**

* Import Data using pandas.
* Encode columns having string values using sklearn.preprocessing.
* Fix feature list for use.
* Split data into training and testing data using sklearn.model\_selection.train\_test\_split in 70% to 30% datasets.
* Check data distribution through histograms using matplotlib for data insight (useful for pre-pruning), as seen below.



* Create classifier with set features and “Loan\_Status” as label using sklearn.tree.DecisionTreeClassifier.
* Change criterion and use gini index and entropy in classifiers.
* Predict accuracy of both classifier metrics and download decision tree as image using graphviz.

**Classifier Parameters:**

1. **Criterion:**

Decides which heuristic to use, offers two heuristics which are “gini” and “entropy”.

1. **Splitter:**

This parameter decides how to search parameters (entries) for split. Default value is “best” while the other parameter being used “random”.

1. **Max\_depth:**

Regularizes the height of the tree which is important as it restricts over-fitting. We are using default which is “None”.

1. **Min\_samples\_split:**

The minimum No. of samples which can be used as nodes. This parameter is used for tree growth restriction.

1. **Min\_samples\_leaf:**

Samples which are to become leaf nodes also used for tree growth restriction. In our case it is “1”.

1. **Max\_featues:**

The number of featues to consider when looking for best split. In our case we have passed feature list to check all features under consideration i.e. “max”.

**Pruning:**

For post pruning scikit learn library uses the extension of DecisionTreeClassifier.cost\_complexity\_pruning\_path in which the concept of alpha is used i.e. the nodes with the weakest links will have the smallest value of alpha and will be pruned first; this in turn increases overall impurity but removes over fitting.

Through overview of histogram distribution of data it was found that the feature “Loan\_Amount\_Term” had recurrent entries of value 360 days both in the case of loan approval as well as loan rejection hence this features was removed. The difference in accuracy whilst using gini index as heuristic was of **1%**.

* Decision tree have also been provided as images.

**Q) Apply a Shallow Neural Network and test different Activation Functions. Also implement SVM on the same dataset and submit a document showing achieved results.**

**Tool of choice:** Anaconda Navigator – Jupiter Notebook.

**Language:** Python 3.

**Libraries used:**

1. Data Import:
   * + - Pandas
2. Numeric Operations / Preprocessing:
   * + - Numpy
       - Sklearn.metrics
       - Sklearn.preprocessing
3. Scikit Learn(General):
   * + - Sklearn.model\_selection.train\_test\_split
       - Sklearn.preprocessing import StandardScaler
       - Sklearn.metrics import classification\_report, confusion\_matrix
4. Scikit Learn(NN):
   * + - Sklearn.neural\_network import MLPClassifier
5. Scikit Learn(SVM):
   * + - Sklearn.neural\_network import SVC

**Steps followed:**

1. Import Data using pandas.
2. Encode columns having string values using sklearn.preprocessing.
3. Fix feature list for use.
4. Split data into training and testing data using sklearn.model\_selection.train\_test\_split in 75% to 25% datasets.
5. Create classifier with set features and “Loan\_Status” as label using sklearn.neural\_network.MLPClassifier for a Neural Network and sklearn.neural\_network.SVC for SVM.
6. Change criterion and use gini index and entropy in classifiers.
7. Change different parameters for both Classifiers and use different activation functions.
8. Predict precision, recall, f-1 score and support of both classifier metrics.
9. Compare Results for best set of Classifier parameters.

**Classifier Parameters for NN:**

1. Different Activation functions used are as follows;
   1. relu
   2. Identity
   3. Logistic
   4. Tanh
2. Learning\_rate was set to 🡪 “adaptive”.
3. Alpha which is penalty parameter is set to “0.01” which by default is “0.001”
4. Maximum Number of Iterations per batch are 🡪 1900 while default were “1000”
5. Hidden Layers 🡪 3 with Nodes per hidden Layer 🡪 10

**Results:**

1. For Relu activation function;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 84% | 56% | 68% | 48 |
| 1 | 83% | 95% | 89% | 106 |

1. For Identity activation function;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 88% | 60% | 72% | 48 |
| 1 | 84% | 96% | 90% | 106 |

1. For Logistic activation function;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 0% | 0% | 0% | 48 |
| 1 | 69% | 100% | 82% | 106 |

1. For Tanh activation function;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 83% | 60% | 70% | 48 |
| 1 | 84% | 94% | 89% | 106 |

**Classifier Parameters for SVM:**

1. Different Kernels used are as follows;
   1. Linear
   2. Poly
   3. Rbf
   4. Sigmoid
2. Gamma parameter which determines the significance of training example is set to “auto”.
3. Class weights are set to 🡪 “Balanced” which determines weights initially according to recurring values or frequency.

**Result:**

1. For Linear kernel;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 95% | 48% | 64% | 44 |
| 1 | 83% | 99% | 90% | 110 |

1. For Ploy kernel;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 47% | 59% | 53% | 44 |
| 1 | 82% | 74% | 78% | 110 |

1. For rbf kernel;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 47% | 61% | 53% | 44 |
| 1 | 82% | 72% | 77% | 110 |

1. For sigmoid kernel;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 | 0% | 0% | 0% | 44 |
| 1 | 71% | 100% | 83% | 110 |

**Overall Conclusion:**

* In Neural Networks results; it was found out that Identity activation function performs best for our dataset and Logistic performs the worst.
* In SVM results; it was found out that linear kernel performs best for given dataset and Sigmoid gives worst results.
* Our dataset was preprocessed in the form of categories constituting 0s, 1s, 2s and onwards. As sigmoid function only return 0 or 1 which is suitable for binary classification and our classification must have overlapping and nonlinear.