EE 559 – COURSE PROJECT – ADULT DATASET

SUBMITTED TO

PROF KEITH JENKINS

Ming Hsieh Department of Electrical Engineering

University of Southern California

SUBMITTED BY

VAISHNAVI KRISHNAMURTHY

vaishnak@usc.edu

3959621652

Ming Hsieh Department of Electrical Engineering

University of Southern California

ABSTRACT

This project aims at finding the best model to predict if the income of an individual is greater than 50K based on the data provided by the UCI Machine learning repository called the '**Adult dataset**' by investigating various pattern recognition techniques. The predictor variables include information such as age, education, occupation, gender, native country and other relevant information of an individual.

The approach included data pre-processing wherein missing data, categorical feature representation, normalization and data imbalance were handled. Also, various pattern recognition techniques- both distribution-free and statistical classification techniques were implemented.

Further, different feature selection and dimensionality reduction techniques such as the principal component analysis, recursive feature elimination etc were used to extract the most representative features for possible improvement in the performance and also to reduce the computational complexity.

Since the dataset was imbalanced, performance metrics like area under the receiver operating characteristics curve and the combined metric F1-score were used to select the best model.

APPROACH AND IMPLEMENTATION

1. PRE-PROCESSING

In order to make the given raw data suitable for classification, pre-processing was performed. The features belonged to one of the following category:

- **Numerical Features (6 in total):** The information provided by these features is in numerical format. The features that belong to this category are age, fnlwgt, education number, capital gain, capital loss and hours per week.
- **Categorical Features (8 in total):** The information provided by these features is in categorical format. The features that belong to this category are work class, education, marital status, occupation, relationship, race, gender and native country.

1.1 Handling missing values: Some of the features (only categorical in this case) for certain data points contain unknown values. It is necessary to substitute these unknown values with the value that best fits the context. The description of the different techniques used to handle the missing data is as under:

- Removing the rows containing missing values: This is in a way the simplest and the best method to deal with missing data as the entire classification depends on the actual dataset i.e. without any assumptions made on the missing values and so the results will be a true depiction of the gathered input data. However, adopting this method can also lead to loss of information in many cases. In our case, there were a total of 622 missing values and we were still left with considerable amount of data (10032 of 10853) after removing the missing values.
- Class independent mode imputation: Since all the missing values in our dataset belong to the categorical features, mode is the best statistical substitute. Other measures like mean or other ways of data interpolation might not produce values which correspond to any of the categories. Hence, all missing value in a particular feature was substituted by the most frequently occurring category of that feature. This intuitively makes sense because the missing value along a particular feature is more likely to contain the category that occur the most. For instance, we replace the missing values in the feature ‘WorkClass’ by ‘Private’ as can be seen from the histogram plot for working class below. However, there is a certain degree of uncertainty attached to it (due to the assumptions made) and this might impact the final classification result either positively or negatively. This method was used for all the three features with missing data: work class, occupation and native country.
- Mode of K- nearest neighbours: In this method, the missing values are imputed using a given set of data points that are most similar (decided by the distance metric) to the data point whose missing values are to be imputed. In this case, all the categorical features were one hot encoded considering

NaN to be a separate category in the dataset. The missing values were replaced with the most frequent categorical value corresponding to the 15 nearest neighbours of that data point. Euclidian distance was used to define the distance metric for the similarity measure. This portion of the code (calculating the Euclidian distance, sorting and computing the mode of every feature in the top k data points) has been written without using any functions or libraries. However, since the results were not better than the previous two methods, there has not been an extensive mentioning of the same.

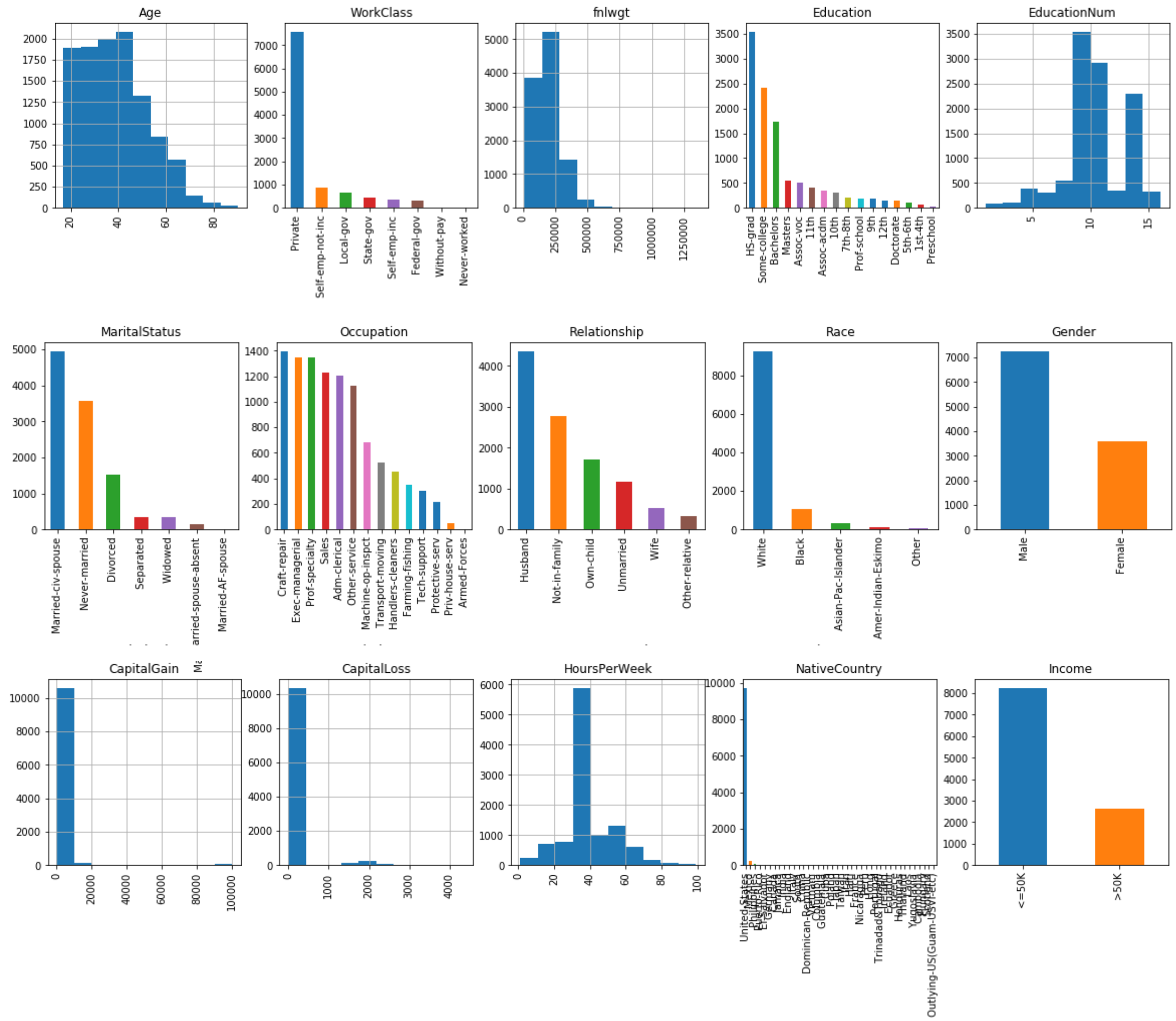
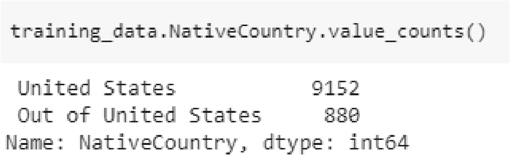


Figure 1.1: Histogram plots for all the features (numerical and categorical)

1.2 Handling categorical features: By analysing the given raw data, the following observations were made.

- The histogram plot for the categorical feature 'Native Country' seems to be interesting. Apart from the fact that there are a lot of categories (40 in total) in this particular feature, the frequency of the category 'United States' seems to be significantly large in comparison with the rest. On further exploration we see that 91.22% of the training data has 'United States' as the 'Native Country' categorical feature. Hence in order to reduce the number of distinct categories considering the significant occurrence of United States, two bins were created - one named as 'United States' and the other as 'Out of United States'.



```
training_data.NativeCountry.value_counts()

 United States          9152
 Out of United States    880
Name: NativeCountry, dtype: int64
```

Screenshot of the code showing the creation of bins

- It is also observed that the same information regarding education is provided in two different features- one in categorical (Education) and the other in numerical (Education Number). Since in the income prediction task, as per our logic, level of education plays a vital role, the following was conducted in order to confirm if one of it is indeed redundant:

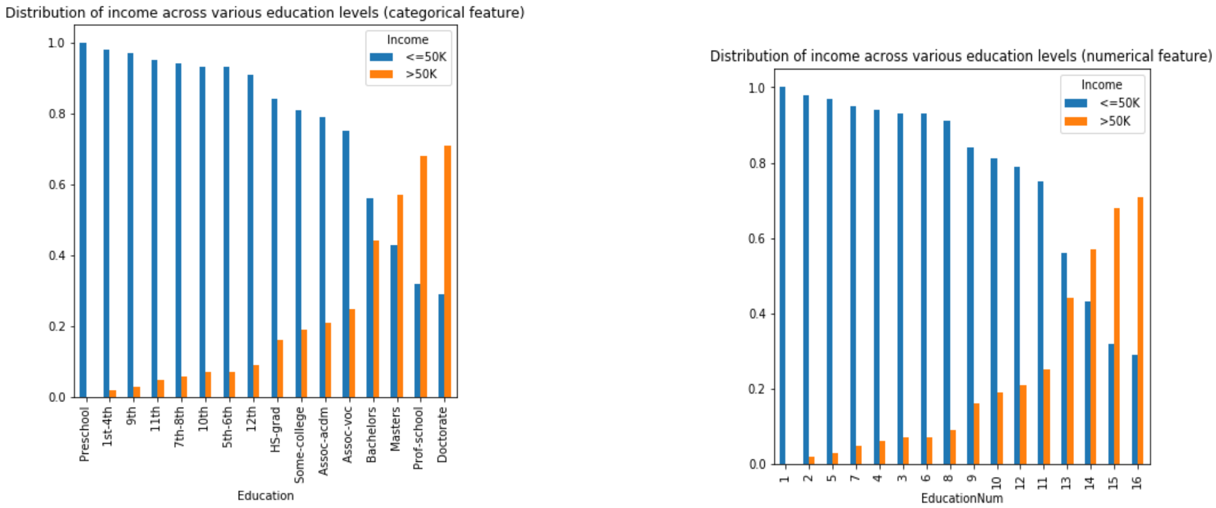


Figure 1.2: Normalized plots of Education (in the left) and Education number (in the right) sorted in the order of increasing income

From the above plots, it is understood that the education number and education indeed provide the same information. Since education number is a numerical feature and it has a logical way of ordering and ranking them i.e. higher number indicating higher educational qualification, we retain this and discard the categorical equivalent of education.

The rest of the categorical features were retained as it is. Also, all the categorical features were encoded to numerical values using one hot encoding scheme. Out of the three features having missing values, the occupation and the working class feature seemed to have a sense of ordering with respect to the target variable (Income). However, label encoding was not used due to the lack of information regarding which category is to be ranked higher than the other. Hence, one hot encoding was used to treat all the categories at the same level.

Note: The resulting training dataset after pre-processing had a total of 49 features.

1.3 Normalization: The numerical features in the data obtained from previous steps need not be along the same scale. Hence, all the numerical features are scaled to zero mean and one standard deviation before further processing. The StandardScalar() function offered by scikit learn library was used in order to achieve this.

COMPENSATION FOR UNBALANCED DATA

For our dataset, it can be seen that 75.92% of the data points in the training dataset correspond to the income lesser than 50K and 24.08% of data points correspond to an income greater than 50K. In order to remove this bias we use SMOTE to balance our datasets. Oversampling on the minority class or under sampling on the majority class can be done in order to strike a balance between the two classes. However, performing under sampling will reduce the number of data points available and hence over sampling seems to be better option. This function synthetically generates the data points for the minority class using the data points belonging to the minority class to achieve a 50-50 balance in the dataset.

However, only for the SVM classifiers, penalized training (cost sensitive training) was used to compensate for the data imbalance and their results have been compared.

FEATURE EXTRACTION AND DIMENSIONALITY ADJUSTMENT

Feature selection is a category of methods that help choose features that contribute the most to the performance of the classifier and to remove the unwanted or redundant features that might unnecessarily increase the computational complexity. This can either be achieved by choosing a subset of features or by transforming the features into a lower dimensional subspace. A description of the feature selection techniques used is as under:

- Select 'k' best features: This is a feature selection method which scores the features in accordance with a metric called f_classif (default) and keeps only the 'k' best features. The SelectKBest() function from scikit learn library was used in order to achieve this. The value of k was chosen such that all features with F-score values greater than 30 were selected.

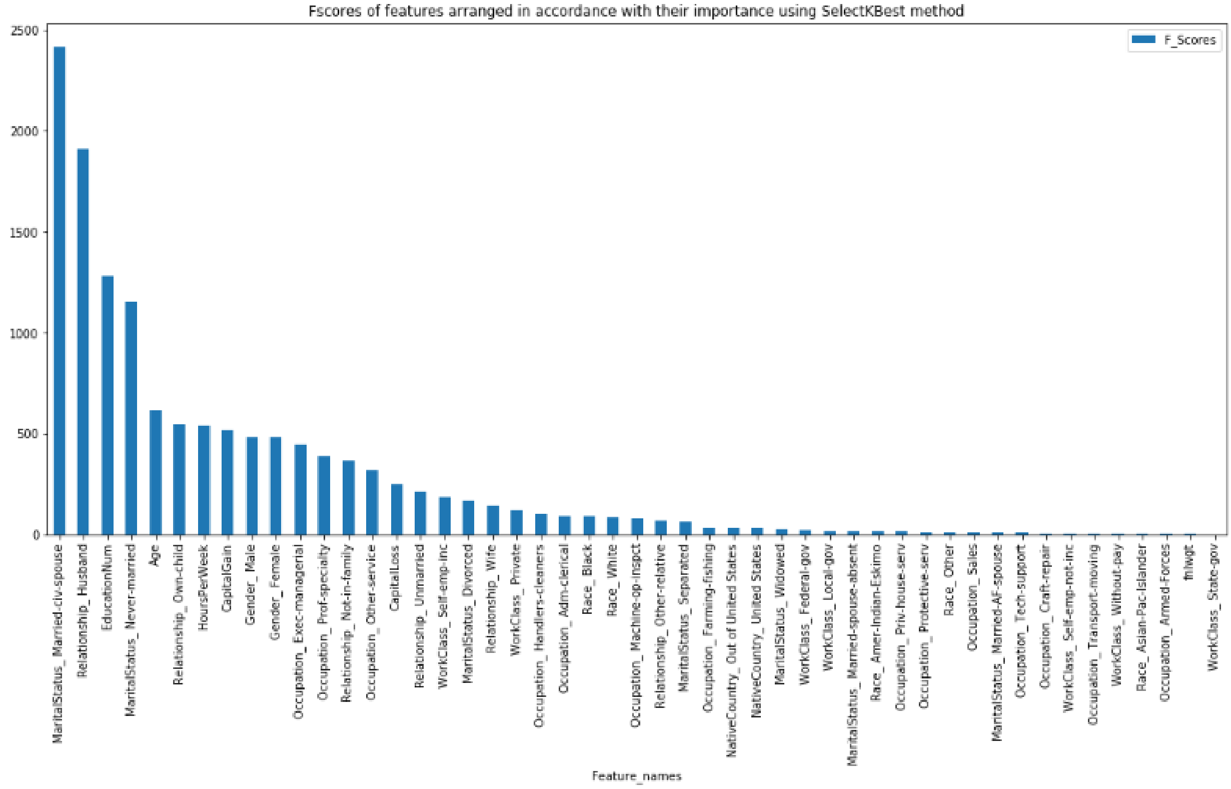


Figure 2.1: F scores of features (49 in total) arranged in accordance with their importance using SelectKBest method

This feature reduction technique was used on the two different imputation techniques (class independent mode and removal of missing rows) mentioned in the pre-processing section.

- Recursive feature elimination (RFE): The goal of this method is to recursively select smaller and smaller sets of features. Initially, the estimator (Random forest estimator was used for this purpose) is trained on all the features and the optimal number of features is obtained by '.n_features_' attribute. The feature with the least importance is discarded and the process is again repeated until the desired number of features to select is eventually reached.

  Function: RFECV() function offered by scikit learn-feature_selection module

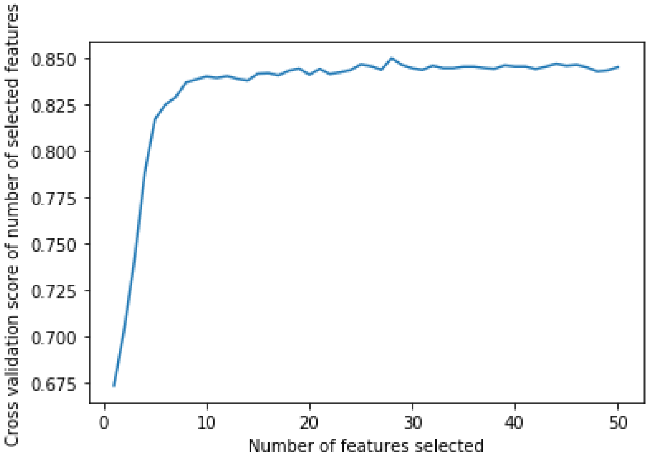


Figure 2.2: Plot of cross validation score with scoring function as accuracy against the number of features [5]

- Principal Component Analysis (PCA): Unlike the previous two methods, the aim of PCA is to transform the features onto a lower dimensional subspace. This is done by computing the eigen value decomposition on the covariance matrix of the input data. This is a widely used dimension reduction technique which is known to produce good results when data corresponding to different classes (since PCA does not take class labels into account) is separable in the lower dimensional subspace.

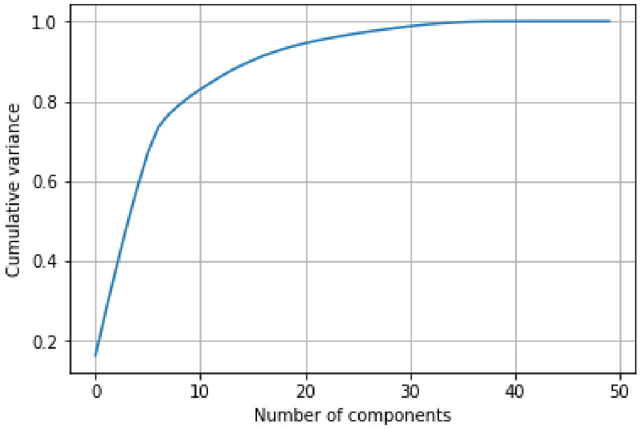


Figure 2.3: Plot of cumulative variance versus number of components using PCA

DATA USAGE

The training dataset contains a total of 10853 data points of which 8240 (75.92%) of them belong to 'lesser than or equal to 50K income' class and 2613 (24.08%) of them belong to 'greater than 50K income' class. The dataset has 622 missing values in a total of three predictor variables which corresponds to 0.06% of the total dataset.

The test data had a total of 5427 data points and the pre processing used on training data was extended to the test data before feeding it to a classifier.

All the classifiers were trained using the entire training data with 5 fold cross validation and run over a range of values (specified against each classifier in the later section) to select the optimal hyper parameters. The classifiers with the optimal parameters were then applied on the test data to determine different classification metrics.

Further, the bigger dataset was used only on the best model to verify if the results were consistent with the ones obtained using the smaller dataset. The bigger dataset contained a total of 32561data points out of which 24720 (75.919%) of them belong to 'lesser than or equal to 50K income' class and 7841 (24.08%) of them belong to 'greater than 50K income' class. The dataset has 1836 missing values in a total of three predictor variables which corresponds to 0.056% of the total dataset.

The test data had a total of 16281 data points and the pre processing used on training data was extended to the test data before feeding it to a classifier.

Baseline performance: The baseline model for this classification task is the accuracy obtained when every data point is assigned the label of the majority class. Hence the accuracy for the baseline is 0.7592.

TRAINING AND CLASSIFICATION

- **Support Vector Machines (SVM):** This is a supervised, distribution free classification technique that focuses on finding a hyper plane that best divides the given data into different classes. The optimal decision boundary is obtained by maximizing the distance between the hyper plane and the points nearest to the plane. These points are called support vectors and variation in the position of these points alters the hyper plane. Since the role of the support vectors is crucial in deciding the hyper plane, they are named so. Two different kernels named the 'linear' and the 'RBF-radial basis function' are used for classification purpose in this particular dataset and the results are compared in the next section.

Function used: SVC offered by the scikit learn-svm module

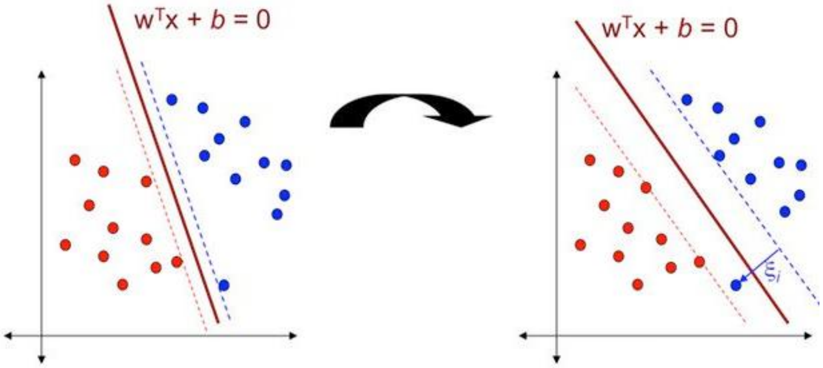


Figure 3.1: An illustration of the SVM wherein the margin is being maximized by compromising on one data point

- **Multi Layer Perceptron (MLP):** This is a class of artificial neural networks which have one or more hidden layers apart from the input and the output layer. The reason behind using this is because the MLP can learn non linear functions unlike the single layer Perceptron which can only learn linear function. In our implementation we find the optimal number of hidden layers for a good classification on the given dataset.

  Function used: MLPClassifier() offered by scikit learn-neural-network module

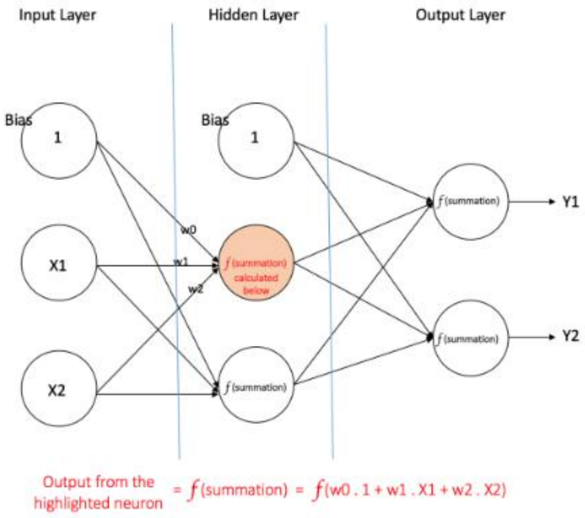

Input Layer
Hidden Layer
Output Layer
Bias
1
Bias
1
X1
X2
w0
w1
w2
f(summation) calculated below
f(summation)
f(summation)
f(summation)
Y1
Y2
Output from the highlighted neuron = f(summation) = f(w0 . 1 + w1 . X1 + w2 . X2)

Figure 3.2: An illustration of MLP with one hidden layer

- **Naive Bayes**: This is a statistical classification technique focussing on determining the probabilities of the features belonging to a particular class. This is done by using the Bayes (conditional) theorem as given below:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

This method assumes that all the features of a given dataset are independent of each other and hence their joint probabilities factor into individual probabilities. The prior probabilities i.e. the probability of occurrence of each class is obtained by using the number of data points belonging to a particular class divided by the total number of data points. This method is known to suit high dimensional data and has a better performance when the independence condition holds true.

Function used: GaussianNB() offered by the scikit learn naive_bayes module- this function assumes Normal distribution for class conditional densities.

- **Decision trees**: This is a supervised learning technique which resembles a tree like structure in its implementation. It starts with all training samples as a root node at the top and at every subsequent layer it considers the feature that provides the maximum information gain. By recursively doing this process it constructs multiple sub-trees over subset of training samples which are classified along the path.

  Function used: DecisionTreeClassifier() offered by scikit learn-tree module

- **Random forest classifier**: This is a supervised learning algorithm that is a widely used as it provides great results without hyper parameter tuning most of the time. The central idea here is that the random forest is a collection of multiple decision trees and that it combines the results obtained from each of those to get a more stable and accurate prediction. Unlike the decision trees where the most representative feature is used to split the node, the random forest uses the best feature among a random subset of features. One advantage of a random forest worth noticing is that it does not over fit the data in most cases if sufficient decision trees are used.

  Function used: RandomForestClassifier() offered by scikit learn-ensemble model

- **K nearest neighbours**: This is a supervised, non-parametric learning algorithm which classifies a given point based on its neighbours. The choice of the ‘k’ becomes very crucial since the data point is assigned to the class of the nearest ‘k’ neighbours. Typically Euclidean norm is considered as a distance metric to decide on how close a data point is to the test data point. Once we get to know such ‘k’ nearest data points, the test data is assigned a label by taking the majority vote from the class labels of the ‘k’ nearest data points.

  Function: KNeighborsClassifier() offered by scikit learn – neighbours module

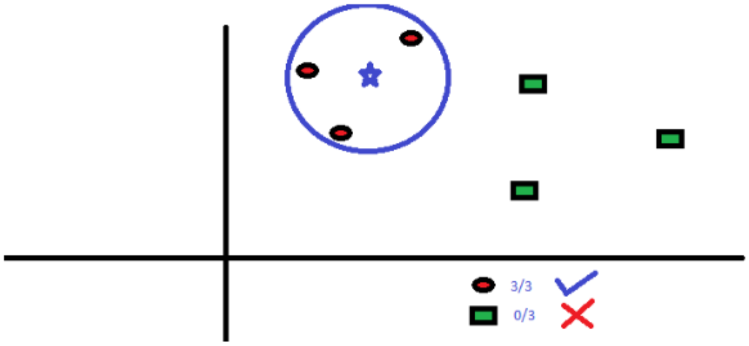


Figure 3.3: An illustration of KNN[4]

COMPARISON, RESULTS AND INTERPRETATION

All rows corresponding to missing values removed									
Classifier	**Training accuracy**	**Test accuracy**	**Precision**		**Recall**		**F1 score**		**ROC_AUC score**
SVM-Linear (weighted)	0.844	0.815	0.87	0.63	0.88	0.6	0.88	0.61	0.8705
SVM-RBF(weighted)	0.847	0.8203	0.89	0.62	0.87	0.68	0.88	0.65	0.8747
Multi-Layer Perceptron	0.8477	0.812	0.82	0.75	0.96	0.35	0.89	0.48	0.8577
Naive Bayes	0.7814	0.80	0.81	0.73	0.96	0.30	0.88	0.43	0.82
Decision Tree	0.855	0.84	0.86	0.75	0.94	0.53	0.90	0.62	0.8565
Random Forest	0.855	0.828	0.83	0.83	0.97	0.38	0.90	0.52	0.8526
K nearest neighbours	0.8425	0.799	0.82	0.67	0.94	0.36	0.88	0.47	0.8036

The two columns under Precision, Recall and F1 score are the respective values for negative (majority class) and positive (minority class). The same holds true for all the other methods implemented in the rest of this report.

SVM Linear: The parameters gamma and the slack variable C were varied in the range (10^{-2} to 10) and the optimal values were obtained through 5 fold cross validation over 100 different combinations of gamma and C. The optimal gamma is found to be 0.1 and the optimal C is found to be 0.0464.

SVM RBF: The parameters gamma and the slack variable C were varied in the range (10^{-2} to 10) and the optimal values were obtained through 5 fold cross validation over 100 different combinations of gamma and C. The optimal gamma is found to be 0.0215 and the optimal C is found to be 4.6415.

Multi layer Perceptron: The number of hidden layers were varied in the range (1,100) and optimal value was obtained through 5 fold cross validation. The optimal number of hidden layers is found to be 10.

Decision Tree: The depth of the decision tree is varied in the range (1,100) and the optimal value was obtained through 5 fold cross validation. The optimal depth is found to be 4.

Random Forest: The number of estimators is varied in the range (1,100) and the optimal value was obtained through 5 fold cross validation. The optimal number of estimators is found to be 85.

K nearest neighbours: The number of nearest neighbours to be considered was varied in the range (1, 25) and the optimal value was obtained through 5 fold cross validation. The optimal number of nearest neighbours was found to be 9**.**

One interesting thing to note here is the behaviour of SVM classifier when it is penalized. For imbalanced datasets, the misclassification penalty per class is generally changed. This is called as the class weighted SVM. The misclassification penalty for the minority class is chosen to be larger than that of the majority class. The plots below show decision boundaries for the normal and the penalized SVM.

Classifier	Test accuracy	F1 score		ROC_AUC score
SVM- RBF (non-weighted)	0.8114	0.89	0.42	0.8612
SVM-RBF (weighted)	0.8203	0.88	0.65	0.8747
SVM Linear (non- weighted)	0.806	0.89	0.39	0.8624
SVM Linear (weighted)	0.815	0.88	0.61	0.8705

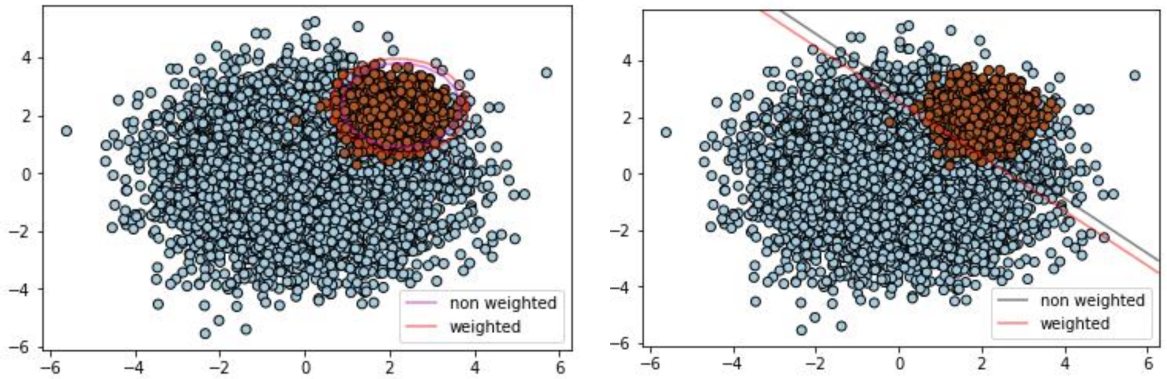


Figure 4.1: Plot of the decision boundaries for weighted and the non weighted SVM using RBF (left) and Linear (right) kernel for the given dataset [3]

As expected, in the weighted SVM case, we observe that the AUC_ROC score, F1 score and other metrics on the minority class has noticeably increased. Hence we conclude that weighted SVM can handle the imbalance in the given dataset reasonably well.

<table>
<tr><th colspan="11">Missing values imputed using class independent mode</th></tr>
<tr><th>Classifier</th><th>Training accuracy</th><th>Test accuracy</th><th colspan="2">Precision</th><th colspan="2">Recall</th><th colspan="2">F1 score</th><th>ROC_AUC score</th></tr>
<tr><td>SVM-Linear (weighted)</td><td>0.8465</td><td>0.822</td><td>0.87</td><td>0.63</td><td>0.90</td><td>0.58</td><td>0.88</td><td>0.61</td><td>0.8730</td></tr>
<tr><td>SVM-RBF (weighted)</td><td>0.851</td><td>0.830</td><td>0.88</td><td>0.66</td><td>0.90</td><td>0.59</td><td>0.89</td><td>0.62</td><td>0.8579</td></tr>
<tr><td>Multi-Layer Perceptron</td><td>0.8517</td><td>0.8231</td><td>0.82</td><td>0.85</td><td>0.98</td><td>0.31</td><td>0.89</td><td>0.45</td><td>0.8621</td></tr>
<tr><td>Naive Bayes</td><td>0.767</td><td>0.802</td><td>0.82</td><td>0.68</td><td>0.96</td><td>0.31</td><td>0.88</td><td>0.42</td><td>0.823</td></tr>
<tr><td>Decision Tree</td><td>0.8558</td><td>0.8496</td><td>0.86</td><td>0.78</td><td>0.95</td><td>0.51</td><td>0.91</td><td>0.62</td><td>0.8768</td></tr>
<tr><td>Random Forest</td><td>0.8567</td><td>0.8253</td><td>0.82</td><td>0.86</td><td>0.98</td><td>0.31</td><td>0.90</td><td>0.46</td><td>0.8512</td></tr>
<tr><td>K nearest neighbours</td><td>0.853</td><td>0.826</td><td>0.85</td><td>0.71</td><td>0.94</td><td>0.45</td><td>0.89</td><td>0.55</td><td>0.8519</td></tr>
</table>

The same procedure as mentioned for the previous method is used to find the optimal parameters for each of the classifiers.

SVM Linear: Optimal gamma and optimal C are found to be 10 and 0.0464 respectively.

SVM RBF: Optimal gamma and optimal C are found to be 0.1 and 4.641 respectively.

MLP: Optimal number of hidden layers was found to be 7.

Decision trees: Optimal depth was found to be 5.

Random forest: Optimal number of estimators is found to be 40.

K nearest neighbours: Optimal number of nearest neighbours to be chosen is found to be 23.

Of all the classifiers that were implemented till now with two different imputation techniques, it can be observed that the imputation of missing values using class independent mode slightly improves the training accuracy while trying to maintain almost the same F1 score and AUC ROC score. However, before selecting the best model, we need to notice the behaviour of the classifiers on the unbalanced dataset. It is obvious that comparing the behaviour of the penalized SVM classifier with the rest is not fair. This is because there have not been any measures taken to address the imbalance in the dataset before feeding them to the rest of classifiers (all the classifiers mentioned above except SVM as class weighted SVM has been implemented). Hence, SMOTE was used to find out if there can be any improvements in the behaviour of the rest of the models. Yet again, as mentioned above, 5 fold cross validation was used to get the optimal hyper-parameters. Their performance after over-sampling using SMOTE is as under:

Over-sampling using SMOTE after class independent mode imputation									
Classifier	**Training accuracy**	**Test accuracy**	**Precision**		**Recall**		**F1 score**		**ROC_AUC score**
Multi-Layer Perceptron	0.8645	0.8173	0.83	0.72	0.95	0.38	0.89	0.49	0.8382
Naive Bayes	0.7991	0.8085	0.81	0.82	0.98	0.24	0.89	0.38	0.8151
Decision Tree	0.8698	0.8142	0.91	0.59	0.84	0.73	0.87	0.65	0.8437
Random Forest	0.8962	0.8245	0.83	0.79	0.97	0.35	0.89	0.48	0.8543
K nearest neighbours	0.8628	0.7750	0.84	0.53	0.87	0.48	0.85	0.50	0.6743

By observing the F1 scores and the AUC_ROC scores, through all the above experiments, it is found that both the SVM and the Decision trees do a pretty good job in classifying the data points belonging to the minority class. By boiling down the options for the best model to one of these two, we tried using different feature selection/extraction techniques to see if there are any significant improvements in the performance. In the table below, the F1 scores and the AUC_ROC scores are reported for the minority class alone i.e. for income greater than 50K case.

The recursive feature extraction method used along with random forest estimator yields a total of 28 features that are considered to be the most representative ones. For the PCA technique, those many components were used such that they capture 95% of variance. The number of components in order to do this was found to be 22. The hyper parameters for all the classifiers mentioned below are tuned using the same procedure mentioned previously.

Feature selection/reduction technique	**Classifier**	**Training Accuracy**	**Test accuracy**	**F1 score**	**AUC_ROC score**
Recursive feature extraction	SVM linear	0.850	0.826	0.57	0.8732
PCA		0.8493	0.826	0.64	0.8799
Recursive feature extraction	SVM RBF	0.8549	0.812	0.38	0.825
PCA		0.8516	0.802	0.64	0.8744
Recursive feature extraction	Decision trees	0.857	0.8396	0.63	0.8463
PCA		0.851	0.7669	0.53	0.6912

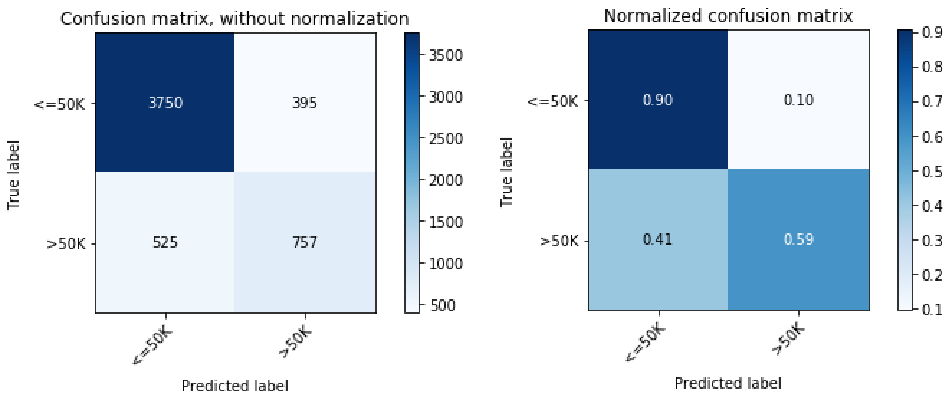


Figure 4.2: Normalized (right) and un-normalized (left) confusion matrices for SVM – Gaussian weighted classifier on the test dataset using class independent mode imputation[6]

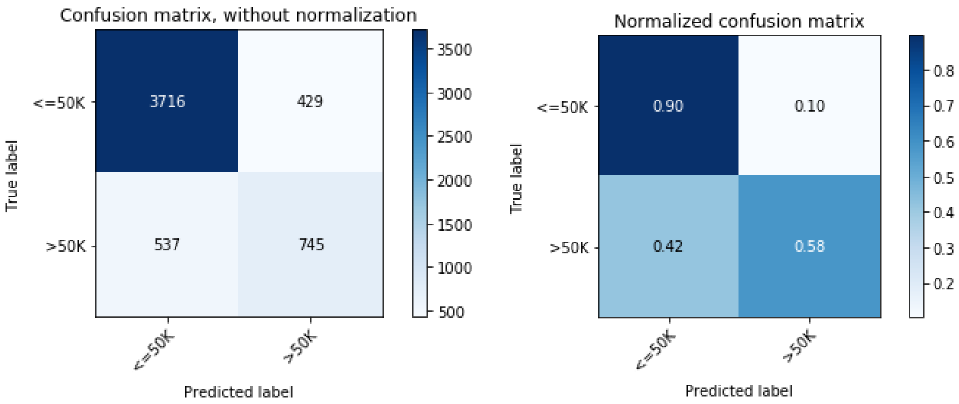


Figure 4.3: Normalized (right) and un-normalized (left) confusion matrices for SVM – Linear weighted classifier on the test dataset using class independent mode imputation[6]

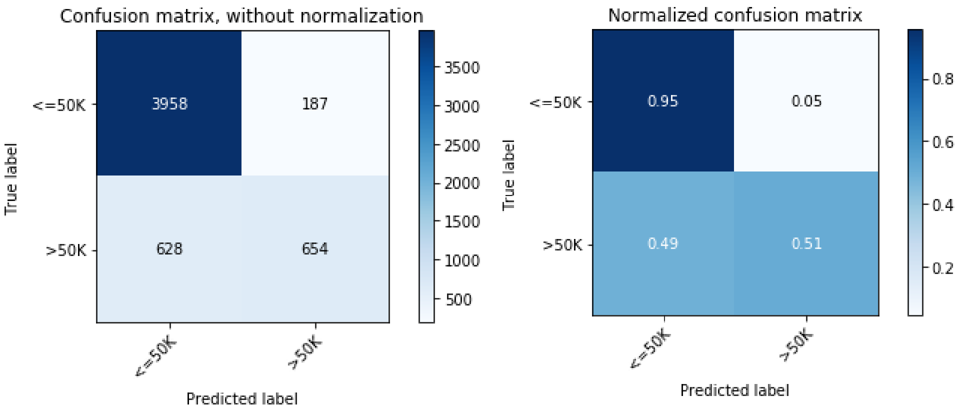


Figure 4.4: Normalized (right) and un-normalized (left) confusion matrices for Decision tree classifier (without SMOTE) on the test dataset using class independent mode imputation

By looking at all the three confusion matrices above, we can conclude that the decision tree classifier performs better on the adult dataset. The decision tree classifier whose optimal depth was obtained on the smaller dataset ‘adult_train_SMALLER.csv’ through 5 fold cross validation was applied on the bigger dataset ‘adult_train.csv’ and ‘adult_test.csv’ to see if the results were consistent. The results obtained were as under:

Classifier	**Training accuracy**	**Test accuracy**	**Precision**		**Recall**		**F1 score**		**AUC_ROC score**
Decision tree	**0.8519**	**0.8523**	**0.87**	**0.78**	**0.95**	**0.52**	**0.91**	**0.63**	**0.88423**

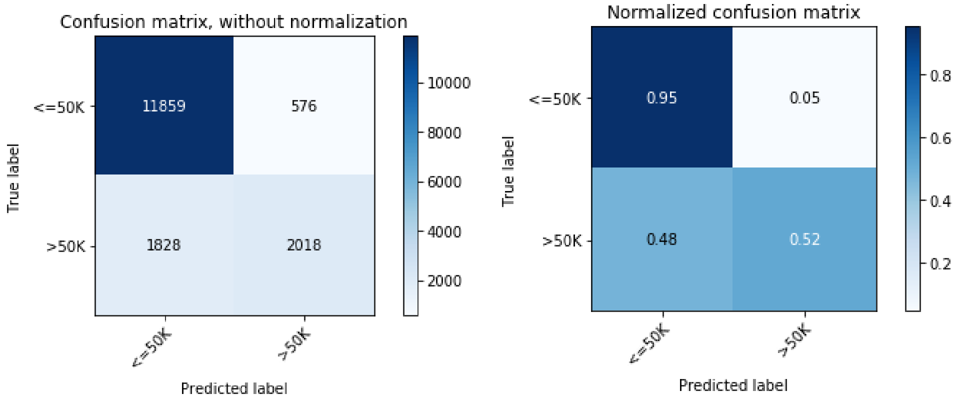


Figure 4.5: Normalized (right) and un-normalized (left) confusion matrices for Decision tree classifier (without SMOTE) on the bigger test dataset using class independent mode imputation

Interpretation of the obtained results:

- Since all the missing values in the training dataset belong to categorical features, it makes logical sense to use the mode imputation.
- The histogram plots help us understand the distribution of various categories in a feature and hence they play a crucial role in guiding towards handling different features.

- As expected, the Naive Bayes classifier does not perform well due to the assumption made on class conditional densities. However, they are simple to implement and serve as a good baseline for the other models.
- It is observed that though with high precision value, F1 score is not impressive for MLP, random forest and KNN, as their recall is poor on the minority class. These values are seen to increase by adopting a data imbalance technique.
- Incorporating different feature reduction techniques by considering around 25-30 most representative features do not significantly vary the performance of the classifiers and only incremental variations can be seen in most cases.
- The values of the performance metrics obtained are robust since cross validation for their estimation renders less variance to it.
- As compared earlier, the penalized SVM performs significantly better than the non weighted SVM. This is because of the greater penalty imposed on every misclassification of the minority class compared to majority class.
- Since all the classifiers implemented above (except the SVMs) were run without addressing the existing data imbalance issue (as can be seen in Table 1 and Table 2), it is quiet natural to see other classifiers not performing on par with the SVMs. However, interestingly the decision trees are able to handle the imbalance in the dataset all by themselves. However, they do not perform on par with the SVMs when it comes to using PCA for dimensionality reduction.
- In all the experiments conducted above we see that the training and the test dataset have their accuracies in the same range. This proves that the classifier is not over fitting the data.
- Further, improvement in the performance of the classifier can be seen with more number of data points (observation made on comparing the results of the smaller training dataset with the bigger one)

CONCLUSION

The Decision tree classifier is found to perform well on adult dataset. This is based on the F1 and the AUC_ROC scores seen in Table 2. However, performing the right set of pre-processing steps and selecting the right set of features plays a significant role in the performance of the classifier. Over-sampling using SMOTE or by using penalized training (in case of SVM) is also seen to improve the performance of the classifiers.

The whole project was implemented on Python. The scikit learn library was used to implement most of the classifiers and the feature reduction techniques.

Description of features in Adult dataset [1]

Predictor variable	Description	Type	Values contained
Age	Age of the individual	Numerical	Integer values between 17 and 90
Work Class	Class of work	Categorical	Federal-gov, Local-gov, Never-worked, Private, Self-emp-inc, Self-emp-not-inc, State-gov, Without-pay
Fnlwgt	Final weight of how much population it represents	Numerical	Integer values
Education	Educational qualification of an individual	Categorical	10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th, Assoc-acdm, Assoc-voc, Bachelors, Doctorate, HS-grad, Masters, Preschool, Prof-school, Some-college
Education Num	Numerical representation of educational qualification	Numerical	Integer values between 1 and 16
Marital Status	Marital status of the individual	Categorical	Divorced, Married-AF-spouse, Married-civ-spouse, Married-spouse-absent, Never-married, Separated, Widowed
Occupation	Occupation of the individual	Categorical	Adm-clerical, Armed-Forces, Craft-repair, Exec-managerial, Farming-fishing, Handlers-cleaners, Machine-op-inspect, Other-service, Priv-house-serv, Prof-specialty, Protective-serv, Sales, Tech-support, Transport-moving

Relationship	Type of relationship	Categorical	Husband, Not-in-family, Other-relative, Own-child, Unmarried, Wife
Race	Race of an individual	Categorical	Amer-Indian-Eskimo, Asian-Pac-Islander, Black, Other, White
Sex	Gender of an individual	Categorical	Male, Female
Capital gain	Capital gain	Numerical	Continuous integer values
Capital loss	Capital Loss	Numerical	Continuous integer values
Hours per week	Average number of working hours per week	Numerical	Continuous integer values
Native Country	Country of origin	Categorical	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad &Tobago, Peru, Hong, Holand-Netherlands
Income	Income level	Binary	>50K, <=50K

REFERENCES:

[1] https://archive.ics.uci.edu/ml/datasets/adult [Online]

[2] Penalized SVM plots from official scikit learn website [Online]

https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane_unbalanced.html

[3] EE 559 discussion and lecture notes

[4]https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/ [Online]

[5]https://scikitlearn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py [Online]

[6]https://scikitlearn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py[Online]