

# CA2: Develop a product release pipeline.

Module Title: Data & Web Mining

Module Code: B8IT108

Module Leader: Ehtisham Yasin

Student Name: Manoj Kumar Thangaraj

Student Code: 10570753

Contents

[CA2: Develop a product release pipeline. 1](#_Toc71031703)

[1. Project overview/scope 3](#_Toc71031704)

[2. Sourcing and cleaning the data. 3](#_Toc71031705)

[3. Exploring and visualising the data 3](#_Toc71031706)

[4. Creating the story 9](#_Toc71031707)

[5. Technology used. 12](#_Toc71031708)

[6. Conclusion 13](#_Toc71031709)

[7. Reflections on learning 13](#_Toc71031710)

[8. References 13](#_Toc71031711)

# Project overview/scope

The main aim of the project is to develop the data mining model on the advertisement dataset to predict if it is an ad image or non-ad image from the UCI Machine Learning repository and the dataset was developed by Nicholas Kushmerick. Further to see if feature reduction is needed or not. Training of the model followed by testing it in numerous ways. Finally evaluating the accuracy of the model on various parameters.

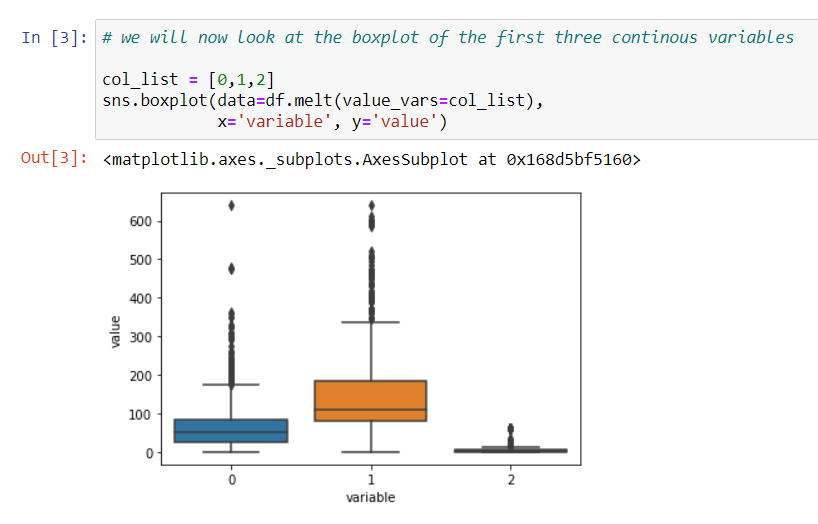
# Platform Selected

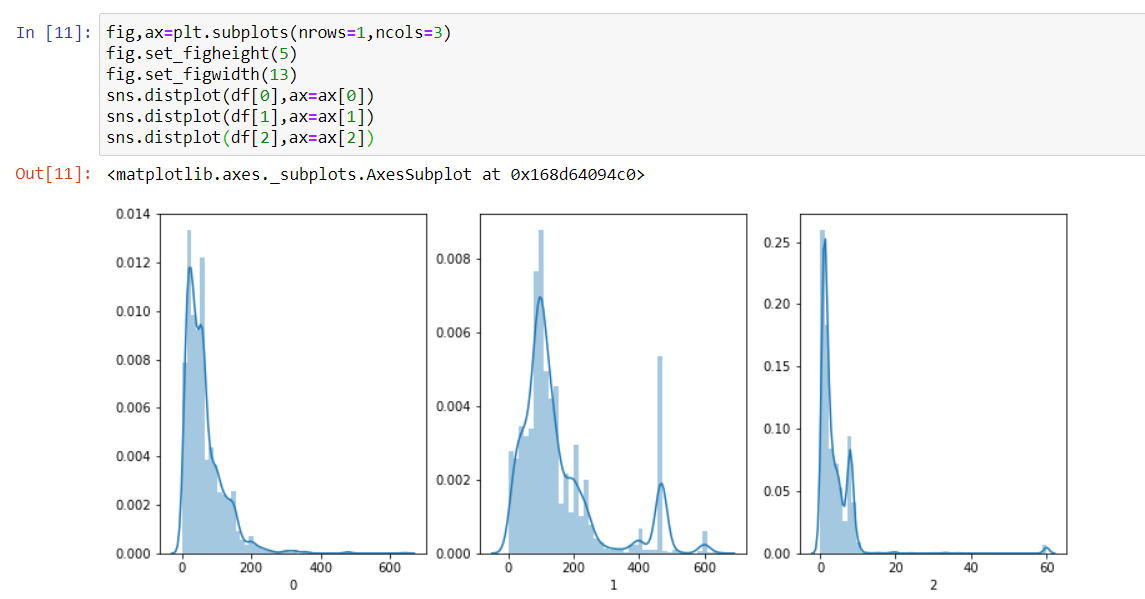
Python in Jupyter notebook is chosen as platform because it is web based interactive computing platform. This platform has various libraires which is easy to work with. It has been chosen also for its best web-browser IDE for python, In-line code execution using blocks, Multiple kernel support, export python code if necessary, etc.,

# Descriptive Analytics

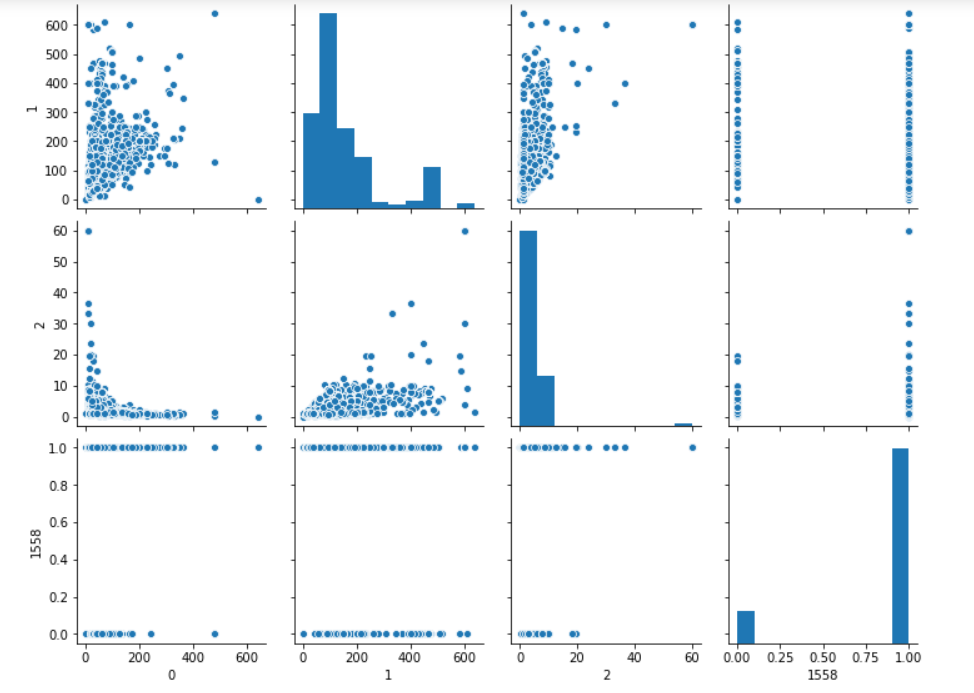
The main reason behind to do this part is to explore the dataset and see how the distributions are and understand if there is any pattern. The file is ad.data format which can be read with pandas using csv reader and the arguments, header as none, non-values as ‘?’ and skip the initial spaces.

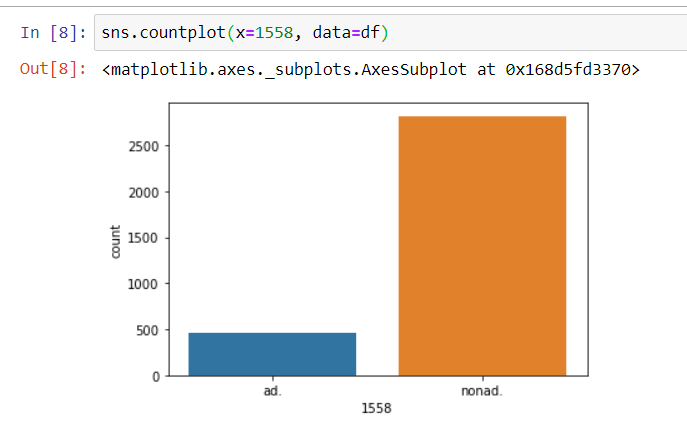
At first, the boxplot for the continuous variables is executed to check for outliers. There were outliers on these variables.



Next, the distribution plot was made to see the distribution of data on these continuous variables and found that they are not normally distributed.

The pair plot was done to check if there is any correlation between the dataset and found that there is high positive correlation between height and width, width, and aspect ratio. The high correlation between the variables may affect the results at the end.



Finally, the count plot is used from seaborn to check for the balance between the classes and found it is hugely imbalanced.

# Data Pre-processing

Data preprocessing is the data mining technique that transforms raw data into an understandable format. It involves primarily checking for the missing values, checking for data types, standardize the data if needed, feature reduction such as PCA and splitting the data. In the case of advertisement dataset, Label encoder is used to convert the string to integer.

Chart

Description automatically generated with medium confidence

The above flow process is the design of the product release pipeline which simplifies the process of developing machine learning models for the advertisement dataset. We have so far defined the problem and brief on the ingestion of data. This section discusses in detail about the preparation of data for the models.

For the missing values,

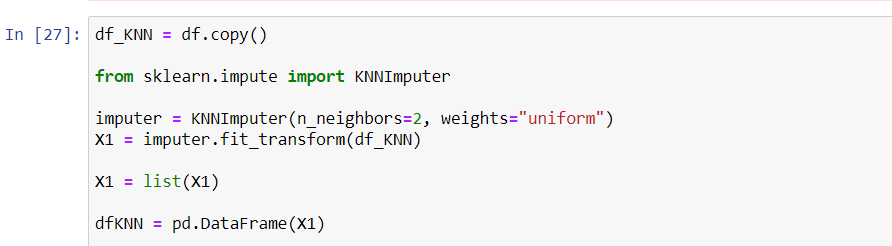
* KNN Imputation method
* Iterative Imputation method
* Replacing with zero / constant values

Diagram

Description automatically generated

**KNN Imputation**

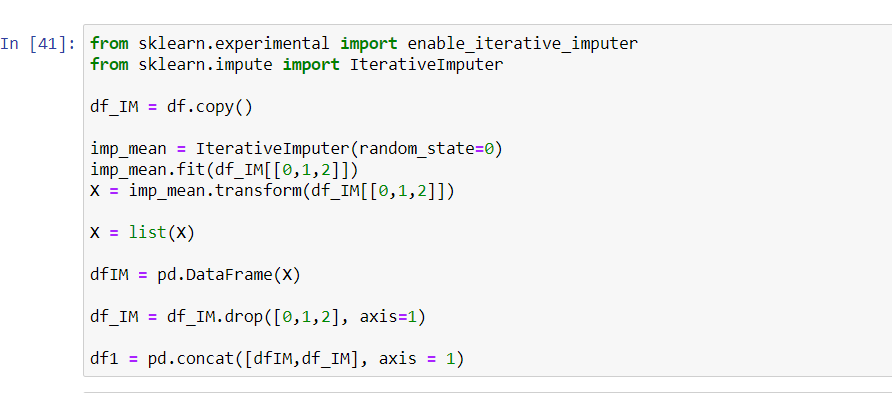
By using this technique, missing values that are marked with NaN values can be replaced with nearest neighbor estimated values. This method often involves selecting the distant measures such as Euclidean and the number of contributing neighbors for each prediction.



New data frame is copied from the label encoded previous dataset. KNNImputer is imported from the sklearn.impute library. In the arguments the number of neighbors is set to 2 though the default is 5, it might overfit the prediction. We assume that the weights of the points in the neighborhood are weighted equally, metrics default to nan\_euclidean and copy to true to create a copy and Boolean add\_indicator default to false. The default one need not be mentioned in the arguments unless otherwise the different one specified which requires definite mentioning in the arguments.

**Iterative Imputation**

Missing values marked with NaN values can be replaced with iteratively estimated missing values. A sophisticated approach involves defining a model to predict each missing feature as a function of all other features and to repeat this process of estimating feature values multiple times. The repetition allows the refined estimated values for other features to be used as input in subsequent iterations of predicting missing values. This is generally referred to as iterative imputation.



The IterativeImputer class cannot be used directly because it is experimental. If attempted to use, the python will throw up an Import error. Therefore, to overcome this problem, enable\_iterative\_imputer library is imported first from the sklearn experimental and then the IterativeImputer. The dataset is copied exclusively for this route. With all arguments with default, it is tried. The parameters are estimator, BayesianRidge being the default, missing values noted as np.nan, sample posterior set to false. Maximum iteration is set 10 by default. Tolerance is set to default. N\_nearest\_features is set to None. Usually it is an int, which means the number of other features to use to estimate the missing values of each feature column. Initial strategy set to mean, which will use initialize the missing values and imputation order is ascending which means from features with fewest missing values to most. Skip\_complete is a Boolean default to False and random\_state to None.

This Iterative imputation is done for only the first three columns because it works well for numeric values only. Since the resulting transform is an NumPy array, it is converted into list and then to dataframe. The old columns are dropped, and imputed ones are concated on axis one. In the column number 4, there are 15 missing values, and the rows are dropped since it is a smaller number of rows. But problem with this part is, there is a probability that some valuable information about this dataset might be lost in this process. Since the missing values are the binary ones, if we go for one of the values, then there is a chance that we represent the wrong information for the class.

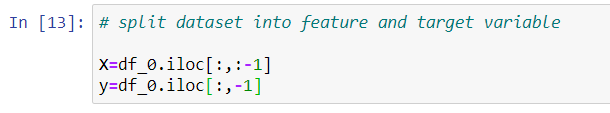
**Imputation with 0/Constant values**

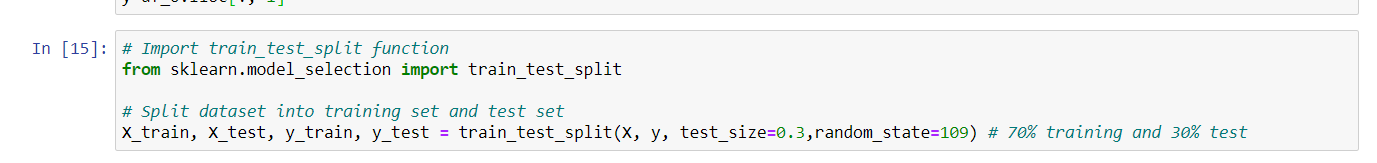
It is yet another simple and most frequent statistical strategy to impute missing values. It works well with the categorical features. The negative side of this problem is it can introduce bias in the data.

For each of the above three methods, three different models were selected and tried to see the performance of the algorithm.

**Test/Train Split**

After trying the different imputation, it is the time to split the dependent or target variables and independent variables. It is done just by creating two different dataframes. One containing all the independent variables and the other one containing the target variable.





The library train\_test\_split imported from sklearn.model\_selection. This will create four different dataframe from training and testing dataframe. Test size is set 30 percent because we need higher percent of data to train. The random state is set to an integer that controls the shuffling applied to the data before applying the split. Stratify to None for this case which is default case.

**Sampling**

There was an imbalance in the dataset.To overcome this issue, there are two ways that are over sampling and under sampling.

Oversampling – Duplicating samples from the minority class

Undersampling - Deleting the samples from majority class.

Both oversampling and undersampling involve introducing a bias to select more samples from one class than from another, to compensate for an imbalance that is either already present in the data, or likely to develop if a purely random sample were taken. But Oversampling is selected for this dataset because of the data available, and to make sure that no information is lost on the sampling process. Therefore, this sampling technique involves selecting random examples from the minority class with replacement and supplementing the training data with the multiple copies of this instance, hence it is possible that a single instance may be selected multiple times.

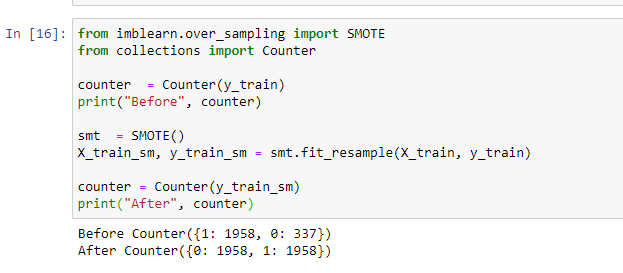
**SMOTE**

Synthetic Minority Oversampling Technique (SMOTE) is an oversampling technique where the samples are generated for the minority class. This algorithm helps to overcome the overfitting problem caused by oversampling technique. It works by selecting examples that are in close in feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbours for that example are found (typically k=5). A randomly selected neighbour is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

Diagram

Description automatically generated

The drawbacks of this model are it tend to create a large of noisy data points in the feature space. The synthetic instances generated are in the same direction i.e., connected by an artificial line its diagonal instances. This in turn complicates the decision surface generated by few classifier algorithms.

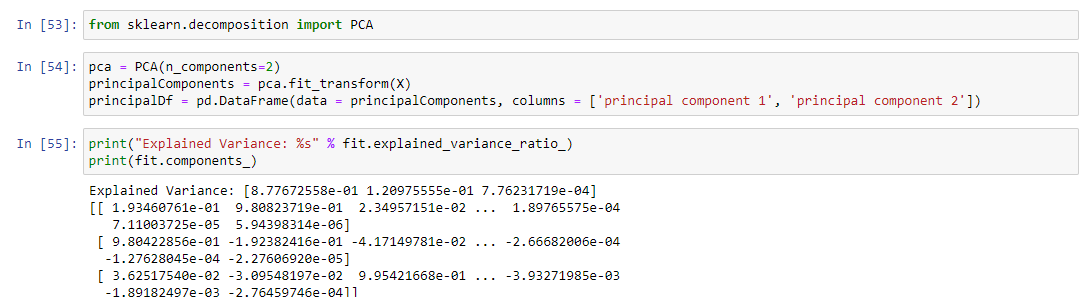


The counter is used from collections library to count the number in each after the smote. In this case, the training dataset is smoted every time before it goes into the model.

**Principal Component Analysis (PCA)**

Once the imputing of the missing values is done, the feature reduction technique is adopted. Since there are more than 1500 features are available, the reduce them into few ones, the PCA is chosen.

Principal component analysis is the feature reduction technique, where number of principal components are chosen at first and the dataset is reduced to the number of chosen components. We can see in most of the PCA technique, more than 80 percent of the variation is explained if the first very few components. So, this technique is chosen to see how the model behaves.



# Model Selection

It is classification problem where the algorithm must learn how to assign a class label to examples from the class domain. Popular algorithms that deal this type classifications are Logistic regression, k-nearest neighbors, Decision trees, Support Vector Machine and Naïve Bayes. Out of all this, Logistic Regression and Support Vector Machine has chosen and instead of Decision tree, Random Forest was chosen. The reason because it is sensitive to noisy data, when we use SMOTE technique, I decided to avoid decision tree and it has an overfitting problem which is greatly overcome by Random forest. Therefore, the selected algorithms are,

* Random Forest
* Support Vector Machine (SVM)
* Logistic Regression

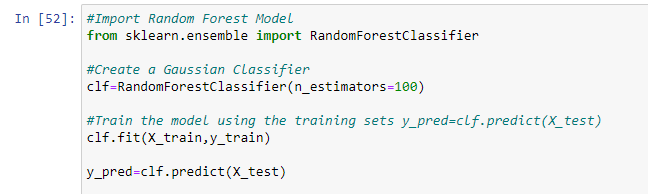
Diagram

Description automatically generated

**Random Forest**

The Random forest classifier like it is name, it has large number of decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become the model’s prediction. The fundamental concept of the random forest is simple but powerful one – the wisdom of crowds. Many relatively uncorrelated models(trees) operating as a committee will outperform any of the individual constituent models. It basically works in four steps,

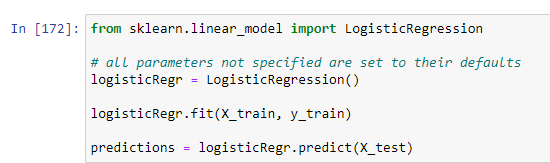
* Selecting a random sample from the given dataset
* Construct a decision tree sample and get a prediction result from each decision tree.
* Perform a vote for each predicted result.
* Select the prediction result with the most votes as a final prediction.



The random forest classifier is imported from the sklearn library, n\_estimators are the number of trees in the forest, the best is chosen out of these 100. Max\_depth is set to default None. Therefore, nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples. Criterion is set to gini index. This measures the quality of the split and max features set to auto and all other variables are set to default.

**Logistic Regression**

Logistic Regression is used when the dependent variable is categorical. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables. It works on the sigmoid function A key difference from linear regression is that the output value being modelled is a binary value (0 or 1) rather than a numeric value based on the result. If the resulting value is below 0.5 then the class belongs to 0 and if the resulting value above 0.5, then the class belongs to 1. Therefore, use case of this algorithm is limited. It is not recommended by life challenging problems.

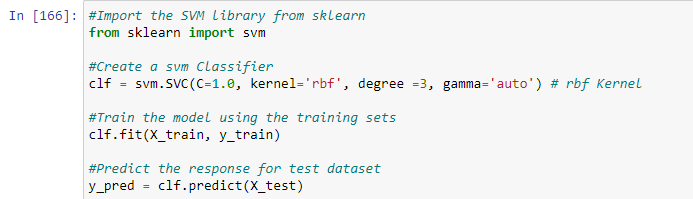


The parameters are set to default. The important parameters are maximum iterations, they are set default i.e., 100. Muti\_class to auto.

**Support Vector Machine (SVM)**

Support vector machine is known for its significant accuracy with less computation power. It can be used in both classification and regression tasks. The main aim of this algorithm is to find a hyperplane in an N-dimensional space (number of features) that distinctly classifies the data points. Hyperplanes are decision boundaries that help classify the datapoints. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

In our case, Support Vector Machine tend to give better results when we PCA transformation with two components.



In our model, C is regularization parameter is set to 1 which is default. Rbf kernel is chosen with degree of polynomial kernel function to 3 and kernel coefficient as auto.

# Model Evaluation

The model is trained and tested in different ways after the three ways of imputation, PCA transformation and the three models selected. The metrics such as Accuracy, precision, recall, F1 score has been selected as one of the evaluation methods and pick two of the algorithms which provides best results and send it through the K-fold cross validation to assess further.

**Precision:**

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

**Recall (Sensitivity):**

Recall is the ratio of correctly predicted positive observations to all observations in actual class Recall = TP/TP+FN

**F1 Score:**

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

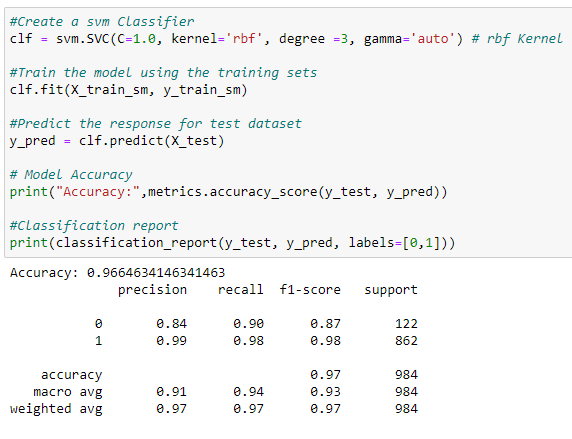
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Imputation Method | | Accuracy | Precision | | Recall | F1-score | Support |
| KNN | Random Forest | 98.47% | Class 0 | 0.97 | 0.91 | 0.94 | 122 |
| Class 1 | 0.99 | 1.00 | 0.99 | 862 |
| KNN | SVM | 93.19% | Class 0 | 0.69 | 0.83 | 0.75 | 122 |
| Class 1 | 0.97 | 0.95 | 0.96 | 862 |
| KNN | Logistic Regression | 97.25% | Class 0 | 0.86 | 0.93 | 0.89 | 122 |
| Class 1 | 0.99 | 0.98 | 0.98 | 862 |
| Iterative Imputer | Random Forest | 93.36% | Class 0 | 0.66 | 0.93 | 0.77 | 120 |
| Class 1 | 0.99 | 0.93 | 0.96 | 860 |
| Iterative Imputer | SVM | 89.48% | Class 0 | 0.55 | 0.76 | 0.64 | 120 |
| Class 1 | 0.96 | 0.91 | 0.94 | 860 |
| Iterative Imputer | Logistic Regression | 95.10% | Class 0 | 0.75 | 0.91 | 0.82 | 120 |
| Class 1 | 0.99 | 0.96 | 0.97 | 860 |
| Replace with 0 | Random Forest | 94.7% | Class 0 | 0.72 | 0.93 | 0.81 | 122 |
| Class 1 | 0.99 | 0.95 | 0.97 | 862 |
| Replace with 0 | SVM | 88.6% | Class 0 | 0.53 | 0.73 | 0.61 | 122 |
| Class 1 | 0.96 | 0.91 | 0.93 | 862 |
| Replace with 0 | Logistic Regression | 94.81% | Class 0 | 0.73 | 0.92 | 0.81 | 122 |
| Class 1 | 0.99 | 0.95 | 0.97 | 862 |

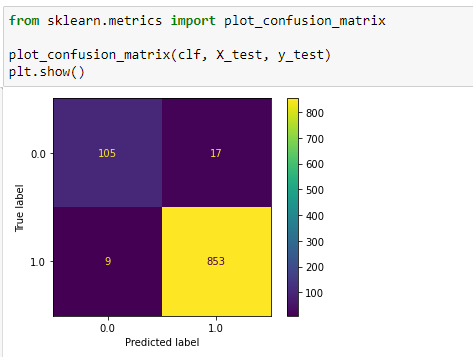
In the above table, the models and scores are listed. Random forest does the best job out of all in the three methods. 100 percent in most of the cases it predicted the class 1 out of the same class present in the dataset. On a maximum of 94 percent predicted values are correct out of all the predicted values. The overall accuracy is 98.47 percent with KNN imputation method. For the class 0 which is an ad, recall of 96 percent in the replace with zero imputation method, this is due to some of the places are replaced with correct binary value. For this case may be random forest would be the best but with KNN imputation method, it was able to correctly predict all the class 1 and 94 percent of the class 0.

SVM yielded a good overall accuracy but recall rate of class 0 in KNN method is 83 percent only which is the highest and in other two method, the rate is nearly the same. This might be due to the higher number features present in the dataset. When the results with PCA are show, this might yield a better result. Class 1 recall was good in all the methods.

Logistic regression yielded the overall better accuracy in all the three methods. The precision was good in all the three methods same as the recall. Since it is a binary classification, it was able to predict very well.

**Model with PCA**

For modelling with PCA, KNN imputation method has been chosen because it performed well in all the four cases. PCA with SVM yielded greater result. This time with two features SVM was able to distinguish the two classes very well. The overall accuracy was 96.66 percent with the recall rate for class 0 is 90 percent and class 1 is 98 percent. This was best prediction in PCA out of the other two tried.



**Confusion Matrix and ROC curve**

Confusion matrix were plotted for each of the model to check for the correctly predicted label and falsely predicted label. It is not visualized in the report. In the code area, it is clearly visualized.

**K-Fold Cross Validation**

From all the above results, Random forest Classifier with the KNN imputation method with no feature reduction and SVM with PCA with KNN imputation method is chosen to send it for K-fold cross validation method.

It is popular technique results in less biased model compare to other methods because it ensures that every observation from the original dataset has the last chance of appearing in training and test set. In the both the cases where random forest with KNN imputation yielded 98 percent overall accuracy with standard deviation of 0.10. On the other hand, SVM with PCA transformed gave us 95.8 overall accuracy with 0.013 standard deviation.

# Conclusion

The advertisement dataset is sent through the data processing techniques to get ready for the machine learning algorithms and three classification algorithms is selected. The processed data is then sent through these models and assessed based on the metrics. The best two of all is taken with developed pipeline it is sent through the K-Fold cross validation method and yielded overall good accuracy rate.

# Reflections on learning

There is various learning on this assignment. Especially in the data processing techniques where its crucial how to select sampling techniques, encoding the classes in the way these algorithms understand, Process the data tailored for the specific model. We have noticed some model performing well on PCA than without feature reduction.

# References

<https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f>

<https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python>

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

<https://machinelearningmastery.com/types-of-classification-in-machine-learning/>

<https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>

<https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>

<https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/#:~:text=SMOTE%20is%20an%20oversampling%20technique,problem%20posed%20by%20random%20oversampling>.