Practical Machine Learning Assignment 2

Name: Melwyn D Souza
Student Number: R00209495
Date: 21/12/2021
Module: Practical Machine Learning
Lecturer: Dr Ted Scully
Course: MSc in Artificial Intelligence

Abstract

For this assignment, I have chosen Australian Weather Dataset, it is a Binary Classification Problem [1] and can be found on kaggle, the target of the model is to predict RainTomorrow. Each instance in the dataset contains information on the weather conditions and a feature RainToday, depending on which the model must predict if it will rain the following day

1 Introduction

Australian weather dataset 'weatherAUS.csv' which contains 22 features such as wind speed, wind direction, humidity, cloudy, rainfall etc and a target feature called Rain Tomorrow (Binary classification problem) which contains values 'Yes' if it will rain tomorrow, 'No' if it wont rain tomorrow. The goal is to design a model which will predict if it will rain tomorrow depending on the input features

The data contains lots of missing values (almost 50%), outliers and categorical values. In this project I have implemented two different pre-processing functions, the first function removes missing values with threshold = 20 (At least 20 features have non NA values). The second pre-processing technique deletes all the rows with missing values. More about the pre-processing techniques will be discussed later.

The pre-processed data is then used to train 8 different models and the best 3 are selected for further validation. Almost all the models have an F1 score accuracy between 0.75 to 0.85. The dataset is then rebalanced using resample techniques and the most important features are retained using Feature Selection Techniques. 8 models are tested again on new processed data. The best performing model is found out, this model is then validated using stratified cross fold validation (10 Folds) and hyper parameter tuning methods using grid search. The research is mainly focusing on rebalancing techniques to improve accuracy and large portion of the report discusses different rebalancing methods

2 Research

My research is mainly focusing on rebalancing the data, this is relevant to the dataset that is selected for this project since the weather dataset is highly imbalanced, class distribution is counted and visualized initially before applying any pre-processing to the data and post pre-processing the data(figure1)

```
No     110316
Yes     31877
Name: RainTomorrow, dtype: int64
```

```
0.0    55514
1.0    15817
Name: RainTomorrow, dtype: int64
```

Figure 1: class distribution counts before pre-processing(left) after pre-processing(right)

The countplot in figure 1 is to visualise the extant of imbalance between two classes.

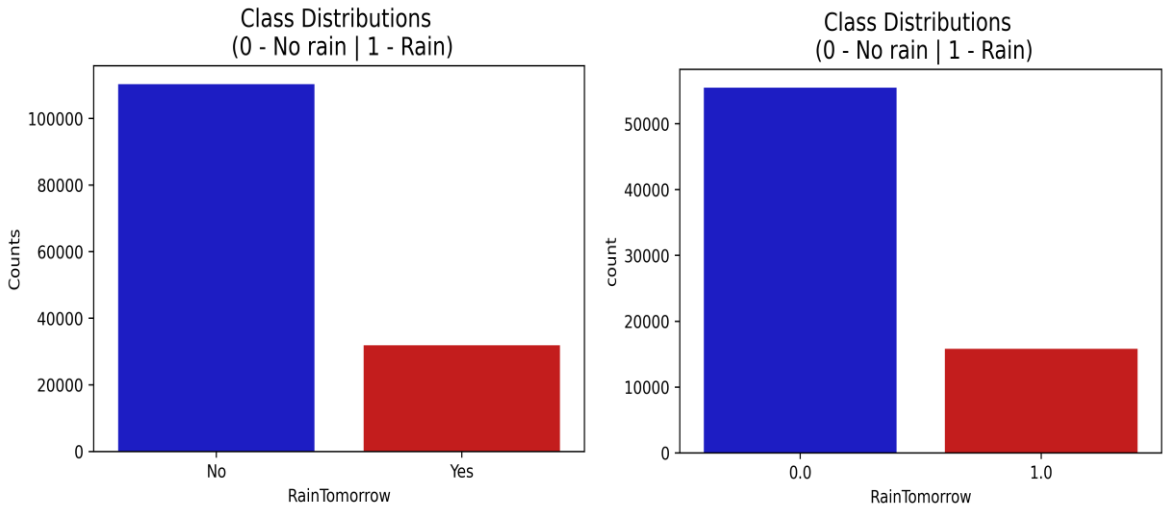

Figure 2: Class distribution before pre processing (left), class distribution after pre processing (right)

Imbalanced dataset might give us very high accuracy since the model is biased towards the majority class. This is a potential problem which is encountered while designing a classification model, therefore, only model accuracy will not be a good indication of the performance, more beneficial metrics to use are confusion matrix and F1 scores

There are several methods used to deal with imbalanced data, I have used 8 different techniques to tackle to imbalance in the weather dataset, we will discuss the investigated methods below, few methods were already discussed in the lectures

1. Random Over Sampling
2. Random Under Sampling
3. SMOTE
4. TOMEK
5. Edited Nearest Neighbours (ENN)
6. ADASYN
7. SVM-SMOTE
8. SMOTE-ENN

2.1 Random Over Sampling:

In this method, the samples in the minority class are over sampled, or the copies of minority class is generated to match with the majority class, it is a non-heuristic method that balances the class distribution by random replication of minority class instances, the main problem with this method is overfitting of the model [2]

2.2 Random Under Sampling:

Here the examples from the majority class are randomly selected and dropped until the count matches with that of the minority class distribution, in the dataset where there are more than 2 classes, the process is repeated on all the classes except that of the class with least examples, the count of all classes is matched to the minority class. This comes with a huge problem, when we delete good data instances, we are also losing information, the instances that are dropped could be carrying very useful information which might have a big impact on the accuracy of model

2.3 SMOTE:

Synthetic minority oversampling technique(smote) is one of the over-sampling methods, the minority class samples are oversampled by creating synthetic samples and not the copies or replications of the other examples. The synthetic samples are created along the line which connects the sample under consideration and any of its K neighbours, the difference between the sample and its neighbour is calculated and then a random number between 0-1 is multiplied to the difference, which results in a new sample. This method forces the decision boundary to be more generic, the visuals of the SMOTE can be seen in the figure 3 [4]

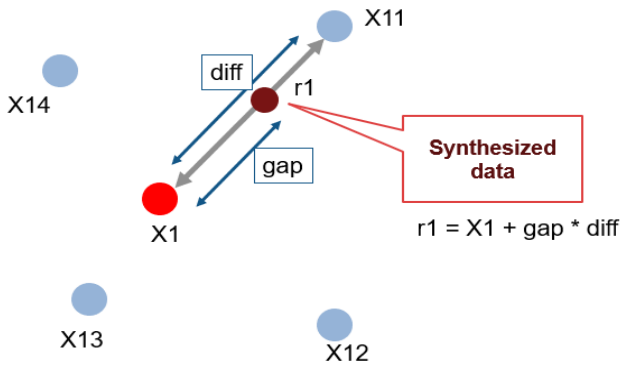

Figure 3: SMOTE calculation and synthetic sample creation [4]

2.4 TOMEK:

Tomek link is an under sampling technique, it focuses on the significant overlap between target classes, as an example consider x to be of class A and y to be of class B, d(x,y) is the distance between two samples, x,y is a Tomek link if and only if x is the nearest neighbour of y or vice versa, the sample in the majority class which is a Tomek link is then dropped. SMOTE and Tomek links can both be combined and used to mitigate the imbalances in the data

2.5 Edited Nearest Neighbours:

This is an under-sampling method which is used to reduce the overlap between majority and minority classes in the feature space, we iterate through all the majority class instances and find its K nearest neighbours, if the class of the observed KNN is different from the majority class then the observed instance is deleted, this will delete all the majority class instances which are scattered in the minority class boundaries

2.6 ADASYN:

It is an oversampling method which was created with SMOTE as a motivation, it created synthetic samples of the minority classes like SMOTE by using KNN methods, the full form of ADASYN is AADAPTIVE SYNTHETIC algorithm [5], it generates samples depending on the density of the minority class, if the minority class samples are spread out in feature space, more synthetic samples are created around that space, whereas if the minority class samples are denser then less synthetic samples are created, as a result ADASYN improves learning in two ways,

1. It reduces the bias introduced by the imbalanced class distribution
2. It will force the learning algorithm to focus on the difficult examples in minority class

In the SMOTE method, equal number of synthetic examples are created for each minority example, whereas in ADASYN it uses density distribution to decide how many synthetic samples have to be generated by adaptively altering the weights of the samples belonging to the minority class

2.7 SVMSMOTE:

In this method, SVM is considered the base classifier, it is an over sampling technique. The goal is to create synthetic or artificial minority class instances around the border or decision boundary of the minority class. The method mainly focuses on borderline instances since they are more prone to be misclassified. If the boundary instances can be classified correctly, the accuracy increases automatically. This method also increases the density of the minority class instances where the density of majority class is low [6]

It is called SVM SMOTE since it used KNN and interpolation as in SMOTE, in addition it also uses extrapolation techniques depending on the sample under consideration and the class of the K nearest neighbours, if less than half of the class is coming from negative class, extrapolate formula is used to create synthetic sample, otherwise interpolate formula is used (like in SMOTE) [6]`

2.8 SMOTEENN:

As the name of the method, it over-samples the minority data samples and brings it up to the majority sample count using SMOTE, it then uses ENN which uses K Nearest Neighbours to select and delete the majority class samples which are overlapping the minority class in feature space. Since we make use of both oversampling and under sampling methods, we will have good balance in the class distribution, also the data boundaries or borderline will be well defined because of ENN[7]

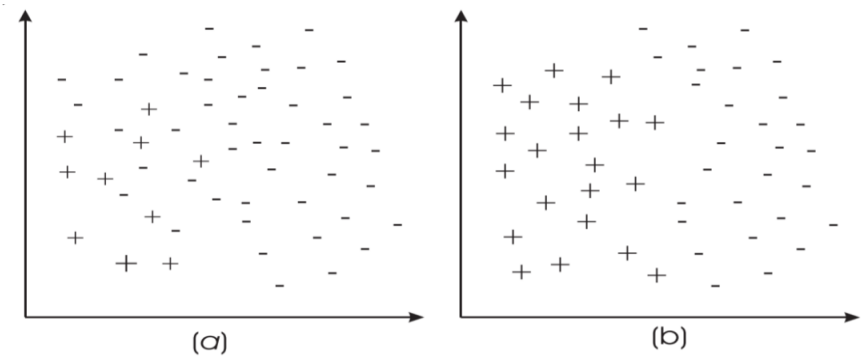

Figure 4: a) Imbalanced data with - majority class overlapping + class in feature space, b) after applying SMOTE ENN the data is balanced and well defined [7]

3 Methodology

3.1 Establishing a Baseline

To establish a baseline, the data is pre processed so that it could be used to train and test several different ML models, the best 3 models are then selected for hyper parameter tuning, the best out of the 3 tuned models is then validated using cross fold validation method.

3.2 Pre-Processing data

The dataset consists 145460 rows and 23 columns, which means, we have 145460 instances and 22 features and 1 target column

The data consists of float, integer, and text values, there are many NA or missing values and outliers which I have dealt with using different pre-processing techniques

I have implemented two functions 1. pre_processing(data) 2. modified_pre_processing(data), these two functions carry out different ways to pre process the data making it ready for the models to validate, the steps followed in the 1. pre_processing(data) is as follows;

1. Drop date and location columns from the dataframe as they are not important features
2. Raintoday, Raintomorrow consists of categorical data (Yes and No) which are encoded as 1 and 0 using LabelEncoder() function
3. Wind direction ('WindGustDir', 'WindDir9am', 'WindDir3pm') have 16 different categorical values (E-East, W-West, NE-NorthEastern etc), these values are encoded using LabelEncoder() so the 16 different categories get 1,2,3….16 assigned
4. The rows with more than 3 features with NA values are dropped (threshold=20 is used)
5. The rest of the rows with missing values are filled with dummy values using Iterative Imputer function, iterative imputer is a process where the observed column is considered as the target and each missing value in the column is predicted as a function of other features, therefore after this step, there won't be any missing value in our data
6. Data is then scaled using StandardScaler(), it is a standard normal distribution which makes mean 0 and variance 1, this helps in improving the accuracy of the model
7. My data consists of many outliers, the figure 5 below shows the boxplot to visualize the outliers prior to applying the Isolation Forest method (not all features are shown), these outliers can mislead our models to predict wrongly which will impact the accuracy of the model, the outliers are dropped with 1% contamination set in Isolation Forest
8. The data is now ready to be used to train the models

The 2nd modified pre processing method is discussed in the experimentation section of this report

3.3 Models

I have tested 8 different models in the pre-processed data when establishing a baseline,

1. Stochastic Gradient Descent Classifier (SGDClassifier()
2. Logistic Regression - LogisticRegression(max_iter = 500)
3. Random Forest Classifier - RandomForestClassifier()
4. Adaptive Boosting - AdaBoostClassifier()
5. Decision Tree Classifier - DecisionTreeClassifier()
6. K Nearest Neighbours - KNeighborsClassifier()
7. Support Vector Machine Classifier - SVC()
8. Naïve Bayes Classifier - GaussianNB()

The function test_all_models() is used to test all 8 models and return the F1 score of all models, 3 best models are selected based on their F1 scores, The best models might be different depending on different pre-processing steps, for example, rebalancing the data using undersampling methods will give SVC as the best performing model, whereas rebalancing using SMOTE gives random forest classifier as the best model

There are different metrics which can be used to evaluate the performance of the model, I have used F1 score, Precision and Recall evaluating the performance of the models

3.4 Experimentation

The 3 models that perform well when testing all models are selected for experimentation, I have carried out following steps as experiments

1. Hyper Parameter Tuning

   This is computationally expensive step, GridSearchCV is used to tune the models and find the best performing parameters of each model, only a few models (3 best performing models) are tuned, and a small search space is used for this step

2. Feature selection

   Random forests are used to select the features, it consists of ~10k decision trees, each tree is built over random extraction of features and observations from the dataset, not every tree will see all the features so there won't be overfitting. It will average the reduction in uncertainty of the feature across thousands of trees and this will help in finding the importance if that feature, by default it uses gini index.

3. Modified pre-processing techniques
   a. Date and Location feature is dropped
   b. Raintoday and RainTomorrow is label encoded using LabelEncoder()
   c. WindgusDir, WindDir3pm and WindDir9am has 16 different categories, these are one hot encoded instead of label encoding, therefore the features increase from 22 to 63 features
   d. The rows are dropped if there is even a single missing value, hence no Imputer is used to fill missing values
   e. Only few columns are scaled using standard scaler
   f. Outliers are detected and dealt with using Isolation Forest method

4. Validation

   We spilt the data into training and test data, but this might now give us the proper accuracy because we might get lucky with the test data and the model accuracy could be very high for that particular split of data, therefore cross fold validation method is used, we have to specify the number of folds (K splits), the data will then be split into K folds out of which K-1 will be used for training the model and 1 fold will be used as a validation fold. The model will then be run K times, on each iteration the validation split will change to the next split, this way the entire data will be used for training as well as testing, the accuracy of the model is then calculated by taking the average scores of all 'K'models

   I have selected the best performing model from above experiments and validated using 10-fold cross fold validation method, the function evaluation_cv in the accompanied code is used for the implementation

3.5 Research

Rebalance dataset

Many different types of rebalance techniques are implemented and tested, these are elaborated in the research part of this report

4 Evaluation

In this section, I will discuss about establishing a baseline, experimentation results and the impact of various rebalance methods on the accuracy of the models

4.1 Part 1 – Establishing a baseline

The data shape prior to pre-processing is (145460,23), after dropping the missing values and filling few missing values with Iterative Imputer, the new shape of the data is (70617, 20)

IsolationForest(contamination = 0.01) is the method implemented to deal with outliers in my code, the boxplot in figure 6 shows the data after dealing with outliers. Isolation forests find outliers by creating many Isolated trees, the data is split repeatedly until an isolated value is found in the feature space.

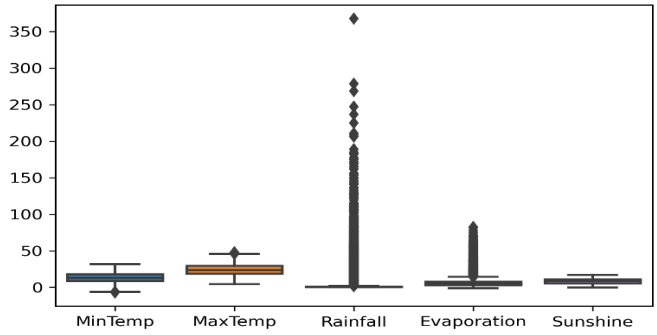

Figure 5: Outlier's boxplot

As we can see, the rainfall feature in the figure above has extreme outliers, I have used contamination of 1% which predicts 1% of the features as outliers

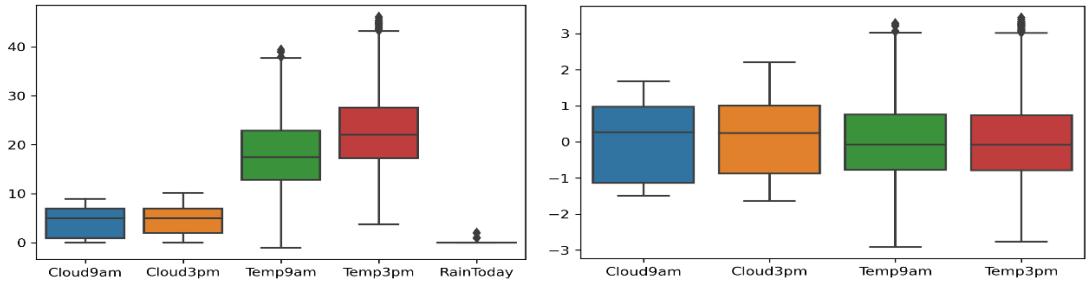

Figure 6: a) pre dealing with outliers b) post dealing with outliers

The models below are run with their default parameters and the F1scores are as listed below in table 1

Model Name	**F1 Score**
SGDClassifier	0.853016143
LogisticRegression	0.856131408

RandomForestClassifier	0.863777967
AdaBoostClassifier	0.85478618
DecisionTreeClassifier	0.79566695
KNeighborsClassifier	0.844378363
SVC	0.861087511
GaussianNB	0.798640612

Table 1: F1 scores of all models

The scores above are excellent, but the problem is, the data is highly imbalanced, Initially I was happy about the F1 scores of all the models, but when printing out the confusion matrix for the best 3 models, we can see that one class is performing very well, whereas other class is performing poorly, the confusion matrix for top 3 models is shows in table 2

RandomForestClassifier	[[10550 483]
	[1441 1650]]
SVC	[[10575 458]
	[1504 1587]]
LogisticRegression	[[10439 594]
	[1438 1653]]

Table 2: Confusion matrix of models run on imbalanced dataset

The minority class [RainTomorrow – yes (1.0)] accuracy is almost 50% whereas the accuracy is highly biased towards the majority class, this is a major problem in every machine learning projects, also a great example of how imbalanced data can give highly optimistic accuracies

These issues are addressed later in research part below where different rebalancing methods are implemented, the balanced data is then used to test model performance

The best 3 performing models from the results above are;

1. Random forest model
2. SVC model
3. Logistic Regression

These 3 models are used for hyper parameter optimization/tuning

4.2 Part 2 - Experimentation

Before hyper parameter tuning the best models, I have modified the pre-processing techniques and dropped features that are not important using feature selection methods

The modified pre-processing f1 scores of all models are listed in table 3 below, initially as discussed above the shape of out dataset was (145460,23), After applying modified pre processing function, the shape of my new dataset is (55291, 65), this is because in the modified function, any row with missing data is dropped (no imputer is used) and few categorical columns are one-hot-encoded, hence the columns have increased from 23 to 65

We can see a slight decrease in the F1 scores of all models compared to the scores in table 1 which made use of different pre-processing techniques

Model Name	**F1 Score**
SGDClassifier	0. 8470024414504024
LogisticRegression	0. 8474545618952889
RandomForestClassifier	0. 8548693371914278
AdaBoostClassifier	0. 8458269282936974
DecisionTreeClassifier	0. 7968170720679989
KNeighborsClassifier	0. 8058594809657292
SVC	0. 8498055882086988
GaussianNB	0. 7393977755674112

Table 3: F1 scored of models with modified pre-processor data

The confusion matrix below shows very poor performance of the top 3 models on the minority class, even if the F1 scores are around 0.80, the accuracy of the minority class is less than 50%

RandomForestClassifier	[[8267 392]
	[1213 1187]]
SVC	[[8275 384]
	[1277 1123]]
LogisticRegression	[[8163 496]
	[1191 1209]]

Table 4: Confusion matrix of top 3 models (modified pre-processor)

Since the performance of the models dropped after modifying the pre-processing methods, I have made use of the first pre-processing function for further experiments

4.2.1 Feature Selection

Feature selection is carried out using Random Forest Classifier method, only the top 10 features are selected, and models are tested with new features, the feature importance scores are listed in the figure below, the best 10 features that are selected is shown in the array at the bottom

```
Feature: 0, Score: 0.04210
Feature: 1, Score: 0.03954
Feature: 2, Score: 0.04919
Feature: 3, Score: 0.03811
Feature: 4, Score: 0.11771
Feature: 5, Score: 0.02799
Feature: 6, Score: 0.05830
Feature: 7, Score: 0.02874
Feature: 8, Score: 0.02761
Feature: 9, Score: 0.03018
Feature: 10, Score: 0.03165
Feature: 11, Score: 0.04724
Feature: 12, Score: 0.15233
Feature: 13, Score: 0.05685
Feature: 14, Score: 0.06545
Feature: 15, Score: 0.03064
Feature: 16, Score: 0.05713
Feature: 17, Score: 0.04042
Feature: 18, Score: 0.04134
Feature: 19, Score: 0.01747
[12, 4, 14, 6, 16, 13, 2, 11, 0, 18]
```

Figure 7: Best 10 features are selected out of 20 depending on the gini index (Random Forest)

Model Name	**F1 Score**
SGDClassifier	0. 8472812234494478
LogisticRegression	0. 8503256867742849
RandomForestClassifier	0. 8572642310960068
AdaBoostClassifier	0. 8489096573208724
DecisionTreeClassifier	0. 7933305012744265
KNeighborsClassifier	0. 8423251203625034
SVC	0. 8549277824978759
GaussianNB	0. 8192438402718777

Table 5: F1 scores after selecting 10 features out of 20 features

RandomForestClassifier	[[10483 556]
	[1460 1625]]
SVC	[[10580 459]
	[1590 1495]]
LogisticRegression	[[10406 633]
	[1481 1604]]

Table 6: Confusion matrix after feature selection

From the above results, the F1 score looks promising after feature selection, since the feature space dimension is reduced from 20 to 10, the computing expense is also reduced by half, also the score is almost same as that of the scores before feature selection, hence since there is reduction in computation time, I have used the top 10 features for further testing

4.2.2 Hyper parameter optimization

Parameter tuning is carried out using the GridSearchCV function, this is a powerful tool for hyper parameter optimization , since there was limited time for this assignment, the search space for parameters is very small

Four best models are selected for hyper parameter tuning and the best parameters of the four models are as mentioned below

1. Hyper parameters for SVC() model
   Best f1 Results: 0.8581577456811101 with parameters: {gamma: 'scale', max_iter: -1}
2. Hyper parameters for RandomForestClassifier() model
   Best f1 score: 0.8572642310960068with parameters: {max_depth: 100, max_features: 'sqrt', min_samples_leaf: 1, n_estimators: 1000}
3. Hyper parameters for LogisticRegression() model
   Best f1 score: 0.8503256867742849 with parameters: {max_iter: 50, n_jobs: -1, solver: 'lbfgs'}
4. Hyper parameters for SGDClassifier () model
   Best f1 score: 0.8423886456727 with parameters: {alpha: 0.0005, max_iter: 600, n_jobs: -1, penalty: 'l2'}

The model selected for cross fold validation is SVC(), this is because it has been performing best out of all 8 models, on average the F1 scores of SVC() with different configurations is 0.85. Therefore, I carry out cross fold validation on SVC using its best parameters as mentioned above

4.3 Part 3: Resampling techniques - Research

Before carrying out cross fold validation, I will apply the researched resampling methods on both types of pre-processed data. Both under sampling and oversampling methods are used to resample the data, figure 8 shows us the imbalance in the class distribution

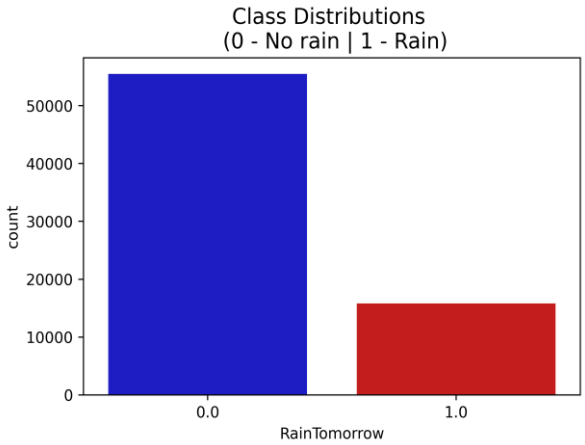

Figure 8: Class distribution imbalance

The class distribution count after implementing 8 different techniques used to resample the imbalanced dataset is shown in the figure 9, we can see under sampling of the majority classes and oversampling of minority classes, these different methods have already been discussed in the earlier section of this report

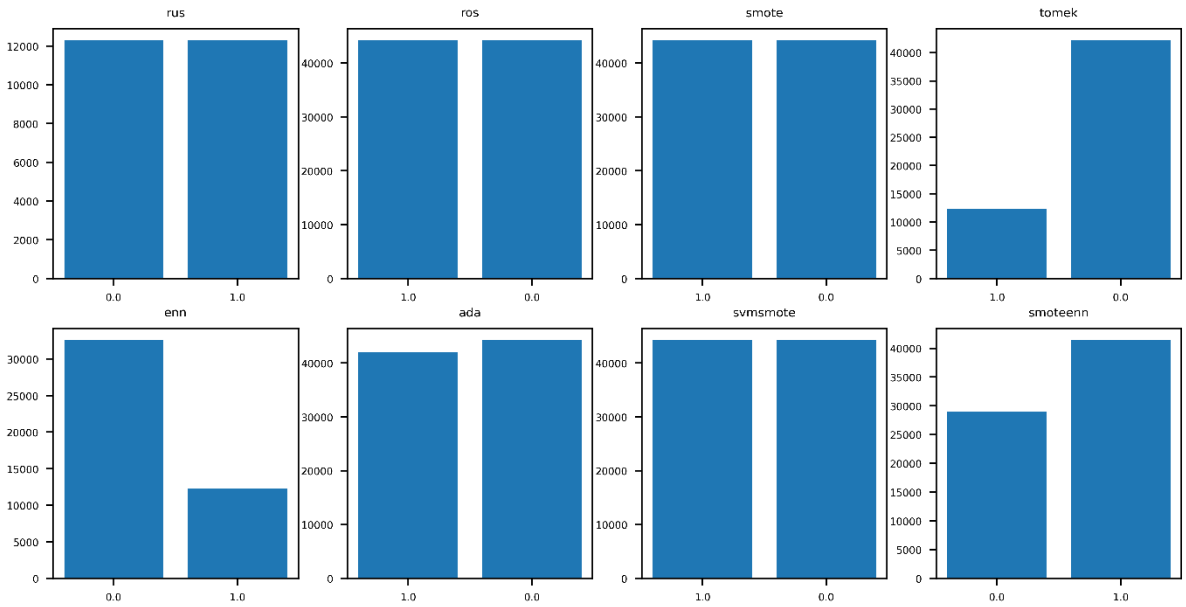

Figure 9: The class distribution after applying 8 types of rebalancing techniques on the imbalanced data

The f1 scores of the 8 models with 8 resampling methods is plotted and shown in the figure 10 and 11, figure 10 is the f1 scores of models run with pre-processed data with first pre-processing function, figure 11 is the f1 scores of models trained using modified pre-process function

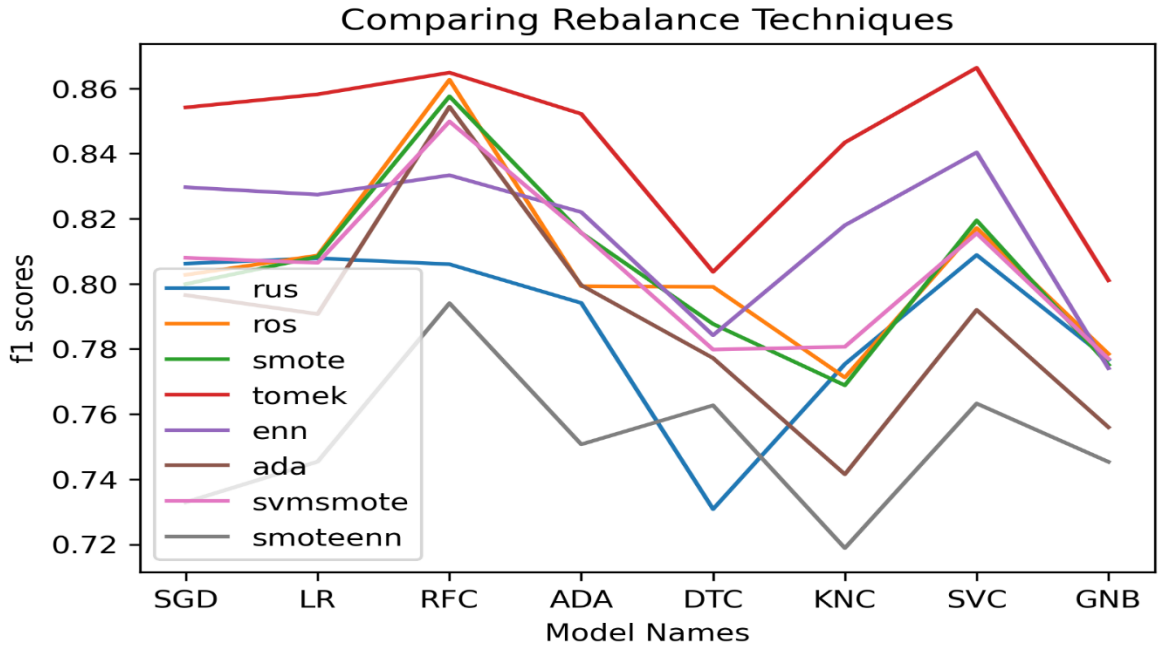

Figure 10: F1 scores of 8 models, with data rebalanced using 8 different methods (pre-process function)

We can see that the model SVC() is performing very well when the data is rebalanced using TOMEK LINKS

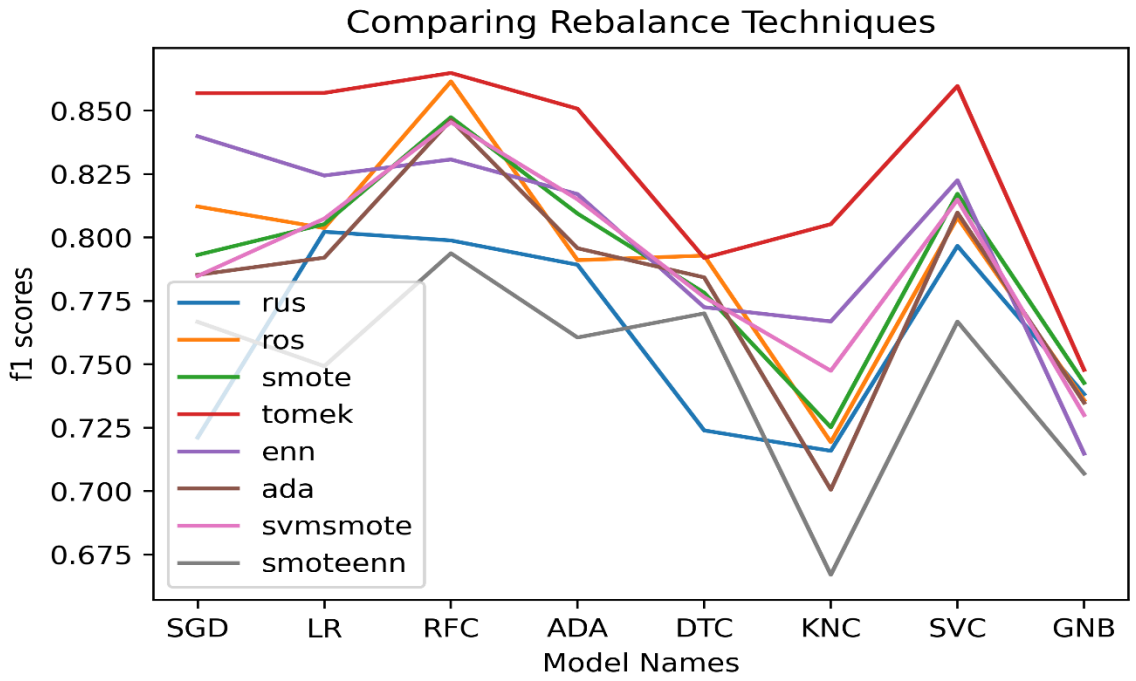

Figure 11: F1 scores of 8 models, with data rebalanced using 8 different methods (modified pre-process function)

I have run all 8 models with the data that is rebalanced using RandomUnderSampler() to show the results and the performance of the model on the minority class

Model Name	**F1 Score**
SGDClassifier	0. 8009770603228547
LogisticRegression	0. 8046587369017275
RandomForestClassifier	0. 8020390824129142
AdaBoostClassifier	0. 7983574058340414
DecisionTreeClassifier	0. 7299631832342113
KNeighborsClassifier	0. 7783913905409233
SVC	0. 8026762956669499
GaussianNB	0. 795737751345228

RandomForestClassifier	[[8765 2259]
	[537 2563]]
SVC	[[8769 2255]
	[532 2568]]
LogisticRegression	[[8894 2130]
	[629 2471]]

We can notice that the f1 scores have been dropped when compared to the earlier results in the report, but if we notice the minority class performance, it has doubled, we now have more than 80%

of the minority class to be predicted correctly whereas it used to be less than 50% when the data was imbalanced

The 30% increase in the minority class accuracy is a very big jump and this model with the reduced f1 score will be a better model when compared to the high false optimistic f1 scores

4.4 Validation

Cross fold validation is carried out on the model SVC() with its best parameters and the rebalance technique 'TOMEK links' is used since the model shows great results with all these configurations

Steps followed before cross fold validating the model are

1. Pre-processing data (1st function)
2. Rebalancing data (Tomek)
3. Feature selection (top 10 features)
4. Hyper parameter optimization (SVC(gamma ='scale', max_iter=-1))
5. Cross fold validation (splits = 6)

```
F1 score is: 0.8658375522782302
F1 score is: 0.8639665419326437
F1 score is: 0.8639515685195377
F1 score is: 0.8606494221243809
F1 score is: 0.8621904237754542
F1 score is: 0.8646119977985691
Mean f1 score (accuracy) is 0.8635345844048027
[[39868.  2372.]
 [ 5067.  7205.]]
```

Figure 12: Cross fold validation of svc() with 'TOMEK LINKS' rebalancing method

Cross fold validation is carried out with 6 folds, the F1 scores at each fold and the average of 6 folds is 0.86 which is a very good F1 score, the model also has accuracy of 60% on the minority class which is not the best but surely is better when compared to earlier configs

The exact same steps are followed but now the rebalance technique used is random under sampling method

```
F1 score is: 0.8048840048840049
F1 score is: 0.796092796092796
F1 score is: 0.8117216117216117
F1 score is: 0.8207570207570207
F1 score is: 0.8048363458720078
F1 score is: 0.8021494870542257
Mean f1 score (accuracy) is 0.806740211063611
[[9937. 2347.]
 [2401. 9883.]]
```

Figure 13: Cross fold validation of svc() with Random Under Sampler rebalancing method

Even if there is a 6% drop in the F1 score accuracy, the minority and the majority class both have ~80% accuracy which is a good machine learning model

5 Conclusion

From all the experiments I have carried out in this project, the SVC() model seems to predict most accurately on the new data instances, the imbalanced data reported high accuracy on other models whereas the models would perform very poorly on minority classes. The issue with imbalance was dealt using different resample methods out of which under sampling methods such as Tomek links which reduce the overlapping of majority class instances over minority classes in feature space have shown excellent improvement in minority class accuracies, also the best results have been obtained when 10 most important features are selected out of all the features and the data is rebalanced using Random Under Sample method, the model would predict 80% of majority and minority class instances right, this seems to be the best configuration I have found out from all the experiments I have performed. In conclusion, pre-processing data, rebalancing and feature selection have had huge impacts on the model performance

5.1 Future work

The data had 1,45,460 instances which was dropped down to 70,000 after pre-processing the data, also when the data is resampled using random under sampler, we were left with only 24,000 instances, with the loss of data, we also lose great amount of useful information. The potential work could be finding out new techniques or models to retain the useful information or to deal with missing values without loss of information, also finding out new ways to increase accuracy of all the class distribution predictions even if there is imbalance in the dataset.

References

[1]https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

[2] Fernández, Alberto, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Salvador García, "*Learning from imbalanced data sets",* Vol. 10, Berlin: Springer, 2018 Oct 22

[3] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[4]https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

[5] He, Haibo, Yang Bai, Et al, "ADASYN" In IEEE International Joint Conference on Neural Networks

[6] H. M. Nguyen, E. W. Cooper, K. Kamei, "Borderline over-sampling for imbalanced data classification," International Journal of Knowledge Engineering and Soft Data Paradigms, 3(1), pp.4-21, 2009.

[7] G. Batista, R. C. Prati, M. C. Monard. "A study of the behavior of several methods for balancing machine learning training data," ACM Sigkdd Explorations Newsletter 6 (1), 20-29, 2004.