# Table of Contents

[Table of Contents 1](#_Toc155175264)

[Introduction 2](#_Toc155175265)

[Methodology 2](#_Toc155175266)

[Input & Output 2](#_Toc155175267)

[Oversampling 2](#_Toc155175268)

[Splitting Data 3](#_Toc155175269)

[Test Harness 3](#_Toc155175270)

[Model Building 3](#_Toc155175271)

[Standardization with Pipeline 4](#_Toc155175272)

[Tuning the Model 5](#_Toc155175273)

[Ensemble Method 6](#_Toc155175274)

[Finalize the Model 8](#_Toc155175275)

[Saving the Model 9](#_Toc155175276)

[Ethical Considerations 10](#_Toc155175277)

[Conclusion 10](#_Toc155175278)

[References 10](#_Toc155175279)

[Appendix 11](#_Toc155175280)

Seattle Weather Prediction

# Introduction

In the previous project, EDA, data visualization, data pre-processing, and feature selection have already been done using the dataset from Kaggle namely weather prediction dataset for Seattle. The same dataset will be applied in this project as well. The result from feature selection in previous work showed that temp\_min has high correlation with other features, but in this project temp\_min will not be excluded because the feature is important for the prediction and there are some methods that will be applied to overcome the overfitting which is balancing the dataset and tuning the hyperparameters that can reduce the risk of overfitting. This project will try to predict the weather based on the features; therefore, the steps are oversampling, splitting data, model building, tuning, ensemble method, finalising and saving the model.

# Methodology

## Input & Output

The first thing to begin this project is defining the input and the output. X is the input and Y is the output.



## Oversampling

Because this dataset is imbalanced which has already been mentioned in previous project, it will start with balancing the dataset with oversampling method using SMOTE () (Brownlee, 2020). It is a method to bring all data to the maximum amount.

A screenshot of a black screen

Description automatically generated

The maximum weather class is 641 and it will increase all classes to 641 with the code below:

A computer screen with colorful text

Description automatically generated



The result showed that all classes are now on par with the maximum class which is 641. Using oversampling for imbalanced data is important to reduce overfitting that can decrease the model performance.

## Splitting Data

After balancing the dataset, the next step is splitting the dataset into training and testing.



The X and y train is the input and output for training, the X and y val is the input and output for testing. The test size is 20% because this dataset is not large enough, so the model needs to be well-trained and putting a lot to test data can make the model to be insufficiently trained because the sample is not enough. The random state will be used to replicate the results, so every time the code runs, the data for train and test is the same.

## Test Harness

For the harness test, which is to estimate the performance using unseen data, K-fold cross validation will be used to test the performance and set the number of splits or k to 10 which means the dataset will be split into 10 parts and the model will be trained on 9 and tested on 1. Accuracy will be the metric to evaluate the algorithm because it gives a quick idea on how much the model predicted right with 100% is the perfect score.

## Model Building

There are many models in machine learning, but for this project, the right model for this problem could not be configured in the first place. There are 6 machine learning models that will be used in this test which are linear algorithm that include Logistic Regression (LR) and Linear Discriminant Analysis (LDA), and nonlinear algorithm that include k-Nearest Neighbors (KNN), Classification and Regression Trees (CART), Gaussian Naïve Bayes (NB), and Support Vector Machines (SVM). The six algorithms above are used because it can handle classification problems and multi-class problems which in this case there are 5 classes to predict for this project.

The code below is used to spot-check the algorithms, and for CART because every time it runs it gives a different result, a random\_state is set to replicate the result. First, is creating an empty list called models to store the model inside it using append function. After that, 2 more empty lists called name and result will be created as well to store the name of the model and the result of the spot-checking. The ‘for’ loop is used to find the accuracy of all models in 1 run, so as long as there is a value inside the models list, it will keep running and here it will run 6 times. Inside the loop, 10-fold cross validation is used to give 10 different performance scores because the n-splits are equal to 10. In order to get the scores, cross\_val\_score have parameters to be filled namely, the model that want to be evaluated, the input and output for training data, the number of folds which already defined in kfold, and the scoring which is using accuracy. Next, is summarizing the 10 scores using mean and standard deviation, mean of accuracy and standard deviation for variance.

The result showed that Classification and Regression Trees have the best accuracy from training which is 80.3% followed by SVM in 77.9% and KNN in 77.1%. The distribution of accuracy will be displayed using box and whisker plot below.

A screenshot of a computer screen

Description automatically generatedA black background with white numbers

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a black screen

Description automatically generatedCART accuracy distribution is higher than the others, but it is wise to also check the table for the value.

A screenshot of a black screen

Description automatically generated

SVM

KNN

CART

Most of the scores for CART ranged above 78 with 85 as the maximum and 75 as the minimum. SVM and KNN, which is the second and third best performing model have most of the score range below 80. Therefore, it is concluded that CART is still the best performing model because it has the best overall accuracy compared to the other models.

## Standardization with Pipeline

Recall in the previous project, the raw data value is varied and may impact the performance of the machine learning models above. Now, standardization will be used as the pre-processing method to transform the value to has the mean of 0 and standard deviation of 1. In addition, pipeline will be used to avoid data leakage which means that the training data slips into the testing data, as a result, the model cannot get a fair estimation on the unseen data. The steps are almost the same as modelling above, first pipeline list will be created to store the model pipeline. The difference is there is a standardization features called StandardScaler() inside the pipeline. 10-fold cross validation will still be used for finding the accuracy of the model with standardized data. Below is the result of the harness test:

A screenshot of a black screen

Description automatically generatedA screenshot of a graph

Description automatically generatedA black screen with white text

Description automatically generated

ScaledCART

The result showed an even better score than before for CART. However, for SVM the accuracy drops drastically and KNN accuracy also drops although not as bad as SVM. Box and whisker plot used to visualise the score again. Overall, the score is better than the other model. The distribution table showed that CART had almost the same score before standardization, but the average score is higher.

## Tuning the Model

Because CART is the best performing model, it will be studied further by tuning the parameters to find whether the accuracy can be boosted to even higher. The code below is used to find the hyperparameters of CART to be fine-tuned.

A black screen with white text

Description automatically generated

The result in Appendix 1.

The hyperparameters chosen for this test namely (Scikit Learn, 2011):

* Max Depth: max\_depth is used to control how deep the tree is allowed to grow in training. Tuning max\_depth is important to prevent overfitting. If the value is too high, trees can be complex, and it can fit noise and outliers in the training data (Kumar, 2021). Therefore, by tuning these hyperparameters, the tree will not be too deep and complex and reduce overfitting. The value of max\_depth is set to [3, 5, 7, 10, 15] to prevent overfitting. The default value of max\_depth is none.
* Min Sample Leaf: min\_sample\_leaf is used to control the minimum number of samples to be in a leaf node. Tuning these hyperparameters can smooth the model, but the main goal is also the same, prevent overfitting. The value of min\_sample\_leaf is set to [1, 2, 3, 4, 5] because it is a medium size dataset, so this value is good to make sure that the leaf has enough information. The default value of min\_sample\_leaf is 1.
* Max Features: max\_features are used to control the number of features to be in each split. Tuning these hyperparameters is used to find the best split. The value of max\_features is set to [1, 2, 3, 4] because this project has only 4 features which are already defined in X which is the input. The default value of max\_features is any, depending on how many features are in the dataset.

The method used to fine-tune CART is Grid search which is an approach to find the best algorithm configuration and combination of the hyperparameters value that have been specified using GridSearchCV. The value of the hyperparameters stored in list. Since the standardised data performs better, it will be used when fitting the data to the model. The main objective of this tuning is to find the best hyperparameters value based on the number that has been set above. The tuning result will be used in validation later. 10-fold cross validation is used to find the accuracy of CART based on the hyperparameters tuning.

Based on the result, it showed that the best accuracy for CART is 80.4% which higher before the tuning process and the hyperparameters configurations are as follows:

* Max\_depth: 15
* Min\_sample\_leaf: 1
* Max\_features: 4

## Ensemble Method

Next method used to improve the performance of the model is using ensemble method. There are 4 ensembles machine learning algorithms to be used here which are boosting and bagging. Boosting method consists of AdaBoost (AB) and GradientBoosting (GBM), and bagging method consists of RandomForest (RF) and ExtraTrees (ET). 10-fold cross validation will be used again for test harness. The standardization data will not be used because the ensemble methods are based on decision trees which are less sensitive or can manage vary data distribution. Below is the accuracy result for 4 ensemble methods alongside the visualization.

A black background with white numbers

Description automatically generated

A screenshot of a screen

Description automatically generated

Based on the result ET and RF perform best with ET 86.8% and RF 86.4% respectively. RF has an outlier, so the table of distribution will be checked. Below is the table of distribution for ET and RF.

A screenshot of a black screen

Description automatically generatedA screenshot of a black screen

Description automatically generated

RF

ET

The distribution table shows that ET has maximum accuracy of 89% and minimum accuracy of 83%, RF has maximum accuracy of 88% and minimum accuracy of 83% which considered as an outlier. Overall, ET is the best performance because they have three 89% accuracy majority of the accuracy is higher than RF and ET does not has an outlier.

Because ET now is the best algorithm, it will be fine-tuned to further boost the performance. Here is the code to find the hyperparameters of ET.



This is the list of ET hyperparameters in Appendix 2.

The method of tuning ET is the same as CART since both machine learning algorithms are decision trees and the difference is that the data used is not standardized like when fine-tuning CART. The value of the hyperparameters is also the same, but for max\_depth 1 more value will be added which is 20 to set the tree to be deeper. Grid search will be used for the tuning and 10-fold cross validation used again for harness test. Based on the hyperparameter tuning, this is the best configuration for ET:

Accuracy: 0.864277 using {'max\_depth': 20, 'max\_features': 4, 'min\_samples\_leaf': 1, 'random\_state': 21}

The accuracy after tuning the hyperparameters is lower than before tuning, which means that the default value for the hyperparameters is far more effective. Also, the best max\_depth value is 20 which is too deep, and it could increase the risk of overfitting although the model can learn more complex.

## Finalize the Model

After concluding that ET is the best model and the result of fine-tuning does not increase the accuracy of this machine learning algorithm, the model will be validated using the test or unseen data. Here is the code used to validate the algorithm.

A screen shot of a computer code

Description automatically generated

First, the model which is the ExtraTreesClassifier will be defined and stored in model with random state of 21 to replicate the value later. Next, the data used to train is X\_train and y\_train and not the rescaled training data because ET is less sensitive to data distribution. In order to predict the class, X\_val is used for the input for validation and y\_val as the output. There are 3 classifications metrics that will be used here which are accuracy score, confusion matrix, and classifications report. Accuracy score is the ratio of how many correct predictions are made by the model. Confusion matrix is showing how many correct and incorrect predictions made in each class. Classification report give more idea on the result namely precision, recall, f1-score, and accuracy. Below is the three metrics result from model validation:

A screenshot of a computer

Description automatically generated

The result of the validation is higher than in training with 88.1% as accuracy and it is concluded that the model performs quite well in handling multi class classification. The confusion matrix shows that:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Number of Test | Correct Prediction | Incorrect Prediction |
| 1 | 111 | 110 | 1 |
| 2 | 137 | 124 | 13 |
| 3 | 137 | 119 | 18 |
| 4 | 139 | 121 | 18 |
| 5 | 117 | 91 | 26 |

The model easily detects the pattern to predict class 1 and is having a hard time predicting class 5. The classification report gives the score based on the confusion matrix and it is showed in the F1-score where class 5 has the lowest f1-score because it has many incorrect predictions and class 1 has the highest f1-score because it has the lowest incorrect prediction.

## Saving the Model

The last step of this project is to save the model using joblib. Joblib is used because according to Brown, models that have a lot of parameters to store can utilise this saving method, and the model used is ExtraTreesClassifier and it has a lot of hyperparameters. Here is the code below alongside with the result:

A computer screen with text

Description automatically generated

A screenshot of a computer

Description automatically generated

Dump and load library is used to store and load the model. The model will be saved with the name of ‘joblib\_finalised.sav’ and the dump function used to save the trained model. The load function is used to load the trained model and make a prediction based on unseen data. Because it used the same test data, the result printed is also the same as the finalised model above showing that the model saved is the correct one.

# Ethical Considerations

Machine learning-based forecasting techniques are not restricted by the laws of physics that control the atmosphere. As a result, it's likely that they will generate misleading results. The best method to overcome this challenge and increase confidence in AI weather forecasts is through collaboration between scientists, forecasters, and forecast users.

# Conclusion

In conclusion, the best machine learning algorithm to predict the Seattle weather data is the ensemble algorithm called Extra Trees Classifier. The result from training reached 86.8%, but after fine-tuning the result was not improved, so the model used the default hyperparameters value for validation and the final accuracy reached 88.1% which is better than training.

# References

Brownlee, J. (2016). Machine learning mastery with Python: understand your data, create accurate models, and work projects end-to-end. Machine Learning Mastery.

Brownlee, J. (2020, August 7). *Multi-Class Imbalanced Classification*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/multi-class-imbalanced-classification/

Kumar, S. (2021, May 31). *3 Techniques to Avoid Overfitting of Decision Trees*. Retrieved from Towards Data Science: https://towardsdatascience.com/3-techniques-to-avoid-overfitting-of-decision-trees-1e7d3d985a09

Scikit Learn. (2011, November 27). *sklearn.ensemble.ExtraTreesClassifier*. Retrieved from Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier

Siva, S.(2023). seattle-weather-predection[Data Set]. https://www.kaggle.com/code/sivaneshansiva/seattle-weather-predection/input

# Appendix

1. Decision Tree / CART Hyperparameters: {'ccp\_alpha': 0.0, 'class\_weight': None, 'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0.0, 'min\_impurity\_split': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'random\_state': None, 'splitter': 'best'}.
2. Extra Trees Classifier Hyperparameters: {'bootstrap': False, 'ccp\_alpha': 0.0, 'class\_weight': None, 'criterion': 'gini', 'max\_depth': None, 'max\_features': 'auto', 'max\_leaf\_nodes': None, 'max\_samples': None, 'min\_impurity\_decrease': 0.0, 'min\_impurity\_split': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'n\_estimators': 100, 'n\_jobs': None, 'oob\_score': False, 'random\_state': None, 'verbose': 0, 'warm\_start': False}