Predict next-day rain in Australia

CS 559: Machine Learning: Fundamentals and Applications

Rohit Pradhan (CWID: 10468609)

Aditi Duggal (CWID: 10460663)

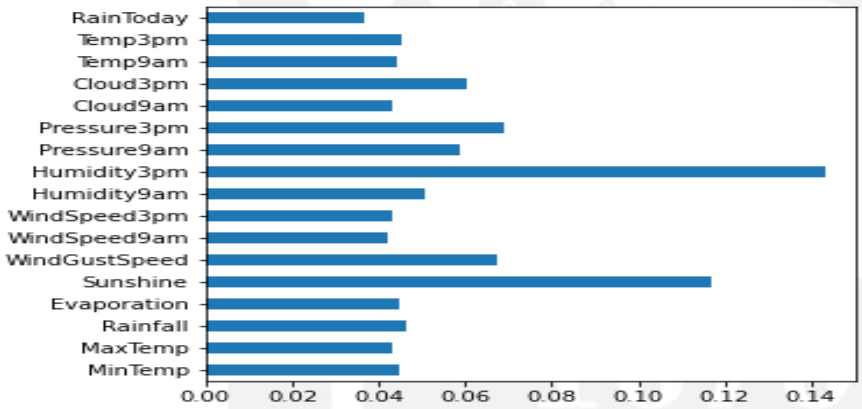
1. **Problem Statement**: Predict next-day rain by training classification models on the target variable "RainTomorrow".

   Source: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

2. **Machine Learning Objective:** We do an exploratory data analysis on the given dataset. This project aims to solve the given problem statement by comparing different Machine Learning Classifiers and choose the best model with respect to evaluation metrics. The best model was deployed for prediction.

3. **Data Cleaning and Exploratory Data Analysis**

   - The NULL values were dropped.
   - The necessary columns were label encoded with appropriate values.
   - Heat Map was used for correlation between features.
   - Outliers were removed based on the quartiles.
   - Data was split into input and output features.
   - Data was split into training and test data (70-30 split).
   - Data imbalance was handled using SMOTE (Synthetic Minority Oversampling Technique).



These are the input features which will be contributing towards model building. The output variable RainTomorrow is:

0: It won't rain the next day.
1: It will rain the next day.

4. **Model Building:** Various machine learning classification models were implemented using scikit-learn library. Algorithms such as logistic regression, support vector, Random Forest, Naïve Bayes, K- nearest neighbors, XG-Boost, Gradient Boost, Ada Boost, CatBoost and Artificial Neural Network were implemented:
   1. Implementation using default values.
   2. Implementation using Grid-Search CV.

5. **Machine Learning Models:**

I. **KNN:** KNN is a supervised classification algorithm. It finds the k-nearest neighbors based on Euclidian distance. It is a lazy learner which means it learns at the time of prediction and is easy to implement as it takes only two parameters (k and distance metric). We find the optimal value for K- neighbors with the help of Elbow Diagram.

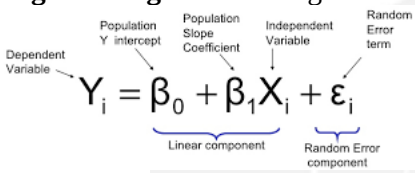
Based on the default parametric values we achieved the model accuracy of 79.74% and this algorithm did not overfit on the dataset.

We tuned Hyperparameter using Grid Search approach, the results for Grid Search with k-folds (CV)=10 are: `BEST PARAMS: {'algorithm': 'auto', 'n_neighbors': 2, 'p': 1}`. The parameter "algorithm" = 'auto' signifies that it will take appropriate values during the fit method. The value for k=2, this was verified with the elbow method and p=1 means it uses Manhattan distance metric. Even though data was pre-processed and hyper tuned we achieved an accuracy of 82.89% which shows that KNN does not works well upon larger dataset.

II. **Naïve Bayes:** It is a supervised classification algorithm based on probability. It assumes that each feature is independent and respectively calculates the posterior probability based on the target class. It assigns the unseen data to the class label which has maximum posterior probability. Even though Naïve Bayes is a weaker model because it makes lot of assumptions on the data, it is giving a better AUC value than KNN.

We observe there is no overfitting for this model. Here we achieved the accuracy of 80.4%.

III. **Logistic Regression:** Logistic Regression is a supervised classification algorithm based on logit function.



ß$_0$ is the regression coefficient equivalent to the y intercept in linear regression. Hence, the name regression.

Based on the default parametric values we achieved the model accuracy of 85.38% and this algorithm is highly overfitted on the dataset with the training accuracy of 100%. Therefore, we need to resolve overfitting issue by iterating the algorithm over 10 cross validations.

Based on tuning of parameters we found: `BEST PARAMS: {'C': 4.281332398719396, 'max_iter': 500, 'penalty': 'l2'}.`
The 'C' parameter is the inverse of regularization. Here we are using L2 regularization as 'penalty' which specifies norm used for penalization. We are iterating the process 500 times= 'max_iter'.

We achieved the model accuracy of 85.30% and the training accuracy was significantly reduced to 79.76% which shows that the issue of overfitting was overcome by tuning the model.

IV. **SVM:** SVM is a classification approach which can be used in both classification and regression problems. It maximizes the distance of the closest samples, support vectors. It forms the hyperplane with the margin of |2b| with respect to the support vectors. This hyperplane helps us to have the generalized model. Kernels in SVM plays an important role which maps the data into lower dimensional and helps to reduce overfitting.

We have implemented SVM with rbf kernel. We obtained the accuracy of 75.83% which is lowest amongst the previous models, but it has a better AUC of 79.67% which shows it classifies false positives and false negatives accurately and hence it generalizes the model.

V. **Bagging and Boosting**: Bagging is a method of merging the same type of models whereas Boosting is a method of merging different types of models. Bagging decreases variance, not bias, and solves over-fitting. Boosting decreases bias, not variance. Bagging techniques like Random Forest, in which trees are grown to their maximum extent, Boosting makes use of trees with fewer splits and converges quickly.

i) Random Forest: It is ensemble technique of classification, widely used in classification and regression problems (CART). It stacks multiple decision trees.
Based on the default values we achieved the model accuracy, training accuracy of 75.83% and 90.41% respectively. It is seen that the model overfits the given data, hence need to hyper tune the parameters.

Based on hyper tune parameters `BEST PARAMS: {'criterion': 'gini', 'max_depth': 4, 'max_features': 'log2', 'min_samples_split': 2, 'n_estimators': 200},` based on the purity of node we found out that 'max_depth' is 4, 'min_samples_split' is 2. With n_estimators = 300, 400 and 500 we get the same accuracy. Therefore, to reduce the computation time the model chooses 200 trees.

$$Gini = 1 - \sum_j p_j^2$$

It calculates the purity of the node. The more gini index decreases more important is the feature.

There is 10% increase in model accuracy after hyper tuning the parameters, the issue of overfitting was overcome as there was a drop in train accuracy.

ii) Gradient Boosting Machine (GBM): A Gradient Boosting Machine combines the predictions from multiple decision trees to generate the final predictions. Here, decision trees are the weak learners.

The nodes in every decision tree take a different subset of features for selecting the best split. This means that the individual trees aren't all the same and hence they capture different signals from the data. Each new tree is sequentially built, meaning it takes into account the errors of the previous trees. Every successive decision tree is built on the errors of the previous trees.

Based on the default values we achieved the model accuracy and training accuracy of 83.85% and 87.80% respectively, and the model doesn't overfit on the given dataset.

Based on hyper tune parameters `BEST PARAMS: {'criterion': 'friedman_mse', 'max_depth': 4, 'n_estimators': 400}`, we have achieved the model accuracy of 85.74%.

iii) Extreme Gradient Boosting Machine (XGBM): XGBoost works on the same principal of GBM. It is an advanced version of GBM with advantages as follows:

- It provides better scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage. It has variety of regularization techniques that reduce overfitting and improve overall performance.
- The XGBM `model` takes care of missing values on its own. The model learns whether missing values should be in the right or left node. Large number of trees might lead to overfitting. So, it is necessary to carefully choose the stopping criteria for boosting.

Based on the default values we achieved the model accuracy 84.02%, and the model doesn't overfit on the given dataset.

Based on hyper tune parameters `BEST PARAMS: {'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 300}`, the learning rate plays an important role in converging the algorithm; we have achieved the model accuracy of 85.84% and AUC of 75.08% which is better than of GBM.

iv) CatBoost: "CatBoost" name comes from two words "**Cat**egory" and "**Boost**ing". "**Boost**" is similar to GBM. So, it is a boosting algorithm that can handle categorical variables in the data. Most machine learning algorithms cannot work with strings or categories in the data. Thus, converting categorical variables into numerical values is an essential preprocessing step. Categorical variables are transformed to numerical ones using various statistics on combinations of features. CatBoost is being widely used is that it works well with the default set of hyperparameters. Hence, we do not have to spend a lot of time tuning the hyperparameters.

Based on the default values we achieved the model accuracy and train accuracy of 86.16% and 93.16% respectively, and the model overfits on the given dataset.

Based on hyper tune parameters `BEST PARAMS: {'iterations': 2500, 'learning_rate': 0.05}`, the learning rate plays an important role in converging the algorithm; we have achieved the model accuracy and train accuracy of 85.58% and 90.42% respectively. The dataset performs best on default parameters.

v) AdaBoost: AdaBoost or Adaptive Boosting is Boosting ensemble model. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances. Boosting is used to reduce bias as well as the variance for supervised learning.

It is a sequential model. Each of the subsequent learner is grown from previously grown learners, meaning weak learners are converted into strong ones. In practice, this boosting technique is used with simple classification

trees or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or another single base-learner.
Based on the default values we achieved the model accuracy 81.38% respectively, and the model doesn't overfits on the given dataset.

Based on hyper tune parameters `BEST PARAMS: {'learning_rate': 1, 'n_estimators': 300},` the learning rate plays an important role in converging the algorithm; we have achieved the model accuracy of 85.20% .

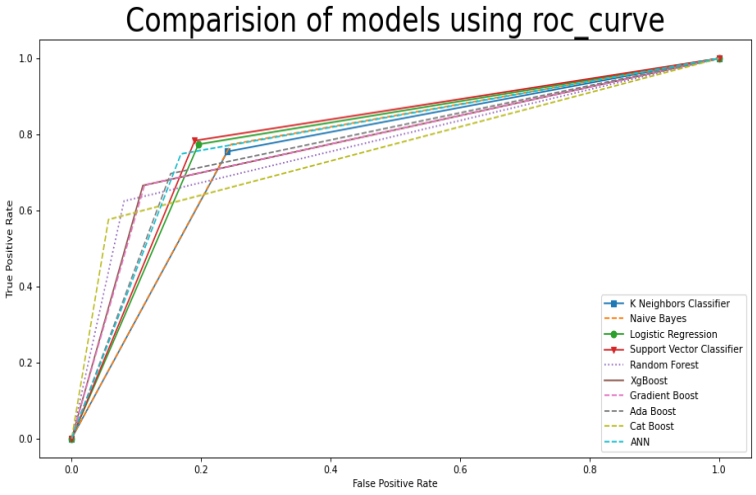
VI. ANN

Artificial Neural Network can be used for classification problems. We have implemented ANN using tensorflow and keras framework. ReLu activation function is used at the input layer and Sigmoidal AF at the output for binary classification. Adam optimizer performs well on the dataset as it works on the principal of Stochastic Gradient Decent which captures the global minima and converges quickly. As we are performing binary classification, we use binary_cross_entropy loss function.

The model is built on 50 epochs and we have achieved an accuracy of 81.29%.

6. Analysis and results:

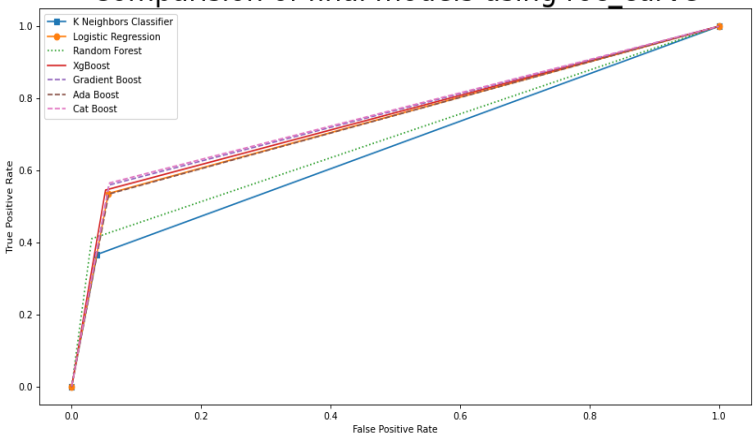
1. The results for default values using accuracy and ROC/AUC evaluation metric are as follows:



	Models	Train Accuracy	Model Accuracy	AUC
0	KNN	0.793188	0.797451	0.757085
1	Naive Bayes	0.799554	0.804052	0.763797
2	Logistic	1.000000	0.853813	0.788999
3	SVC	0.768264	0.758353	0.796678
4	RandomForest	0.904124	0.758353	0.771850
5	XG-Boost	0.879242	0.840256	0.777789
6	GradientBoost	0.878016	0.838530	0.777171
7	AdaBoost	0.838632	0.813750	0.771972
8	Catboost	0.936138	0.861684	0.759473
9	ANN	0.791725	0.812887	0.789914

From the results we conclude that, many of the classifications models are prone to overfitting. To overcome the issue of overfitting we used Grid-Search CV for hyper parameter tuning.

2. The results for hyper tune parameters using accuracy and ROC/AUC evaluation metric are as follows:



	Models	Train Accuracy	Model Accuracy	AUC
0	Logistic	0.797646	0.853052	0.663576
1	RandomForest	0.805670	0.845130	0.739523
2	KNN	0.923210	0.828983	0.689582
3	Catboost	0.904166	0.858079	0.746451
4	XG-Boost	0.897673	0.858485	0.750791
5	GradientBoost	0.899415	0.857469	0.738379
6	AdaBoost	0.875313	0.852036	0.753229

From these results we can conclude that, the train accuracy of models is reduced, and hence the issue of overfitting was resolved. It was also seen that there was significant increase in the model accuracy after tuning each model.

7. Model Selection and Deployment

The XG-Boost model was deployed in Heroku using Flask.

Flask is a web application framework that gives users a platform to develop web applications. Also, it provides the necessary tools and libraries that allow us to build a web application.
Heroku is a cloud platform as a service (PaaS), that enables developers to build, run, and operate applications entirely in the cloud.

The link for our deployed application: https://rainfall-prediction-aus.herokuapp.com/

Sample input values for prediction:	'It will not rain tomorrow'="0"	'It will rain tomorrow'="1"
Minimum temperature	23.6	9.4
Maximum temperature	40.4	16.0
Rainfall	0.6	1.5
Evaporation	11.8	1.0
Sunshine	12.2	2.7
Wind gust speed	54.0	46.0
Wind speed at 9AM	9.0	17.0
Wind speed at 3PM	11.0	20.0
Humidity at 9AM	42.0	78.0
Humidity at 3PM	17.0	67.0
Pressure at 9AM	1008.4	1020.3
Pressure at 3PM	1005.0	1017.8
Cloud at 9AM	1.0	7.0
Cloud at 3PM	2.0	7.0
Temperature at 9AM	29.9	12.9
Temperature at 3PM	38.7	15.3
rain today	0.0	1.0

8. Results and Discussions:

We have successfully implemented all the machine learning algorithms taught in the coursework. Working of each algorithm was studied and results were carried out on the given Dataset. We can see how parameter tuning is a tedious job and plays an essential role in improving the model accuracy. Even though it is time consuming task, we need to work on tuning the parameters for accurate results. After working on 10 algorithms, we found XGBoost outperforms other algorithms with a model accuracy of **85.584%** and area under the curve of **75.07**.

Furthermore, to conclude our results, we can say these results are not universal but specific to the problem statement. We could choose the best model depending on the study case. Hence, model selection depends on 3 factors:
1.Business problem
2. Dataset
3.Requirements of stakeholders.