Rain Prediction model

Chand Sheikh

Abstract Rain prediction can help people to plan their work and many industries can take advantage of rain prediction to improve the overall workflow. With the current model, we can predict the rain for the next day with more than 84% accuracy

1. Introduction

This dataset contains daily weather observations from numerous Australian weather stations. The target classification variable "RainTomorrow means", did it rain tomorrow or not. Rain prediction could be very useful for multiple industries and can save millions of dollars, possible industries that can use this model. Majorly the model can be used for bad weather avoidance and planning the crops for agriculture.

- Air traffic
- Marine:
- Agriculture
- Forestry
- Utility companies
- Other commercial companies
- Military applications

The dataset is popular and used in multiple research project, few links are given below.

- https://library.dbca.wa.gov.au/static/FullTextFiles/923992.pdf
- https://www.biogeosciences.net/10/6545/2013/bg-10-6545-2013.html
- https://mpra.ub.uni-muenchen.de/47787/

Dataset details (Columns):

Date, Location, MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustDir, WindGustSpeed, WindDir9am, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm, Temp9am, Temp3pm, RainToday, RainTomorrow

2 Research (Feature Selection)

Feature Selection is an important aspect of machine learning model building as this can affect your results. Independent features/Irrelavent features can have an effect on the overall result. Irrelevant features can skew your data too.

In order to research, I have tried multiple techniques. The list of techniques the listed below

- Univariate Feature Selection
- Greedy with RFECV Feature Selection
- Chi-Square Feature Selection
- Correlations Feature Selection

2.1 Univariate Feature Selection selects the best features based on statistical tests, each feature is compared with the target feature, we don't consider the other feature in these statistical tests. Only the feature and the target feature are used in the analysis. As no correlation is used for tests, this method is called a univariate feature selection. Univariate Feature Selection will give scores for every feature, more the score, better is the impact of the feature to predict the target variable. If a feature has very low score, It possible that we can build better prediction model without the feature.

2.2 **Greedy with RFECV Feature Selection** recursive feature elimination (RFE) selects the best group of features to predict the target variable, as the name suggests it will recursively remove features and check the accuracy and then select the best subset. We have to pass an estimator model to RFE to perform the prediction. Usually RandomForest estimator is used with RFE. We used RFE with and without cross-validation (CV) and both have given us same result.

2.3 **Chi-Square Feature Selection** takes a different approach and checks the independence of the features with the target variable, if the features are independent and don't affect the target class, then there is no point in keeping the feature in the model. The chi-square will help you check the independence values of the feature. If the feature is independent, remove it. Please take a note that Chi-Square is sensitive to small frequencies in cells of tables.

2.4 **Correlations Feature Selection** [1] checks for correlations, it will help you to understand the highly co-related and low correlated features. In general, it's better to remove the features with correlation more than 0.9 or less than correlation 0.1.

3. Methodology

3.1 pre-processing steps

3.1.1 **Dealing with Missing Values:** The data had a couple of features with approx. 50% of the features missing, so I dropped those features. And then removed the rows with none values. In this process, only approx. 5% of the rows got lost.

3.1.2 **Dealing with Outliers:** I have individually checked features and plotted then using boxplot and identified the outliers and removed them. If there are clusters of data outside the whisker, I have removed them. I only removed the which were unclustered and outside the whiskers

3.1.3 **Handling Categorical Data:** I have changed bi-variate categorical data (Yes/No) to 1 and 0 respectively.

3.1.4 **Scaling Data:** I had a couple of textual features, so I have scaled the data to make it numerical feature.

3.1.5 **Handling Imbalance:** I have checked the target class for the imbalance, The data was imbalance, with 20% of rows with one category and 80% of the rows in the other category. I have used the confusion matrix to check the effect of the imbalance data and tried the SMOTE method to balance data and checked the effect using confusion matrix. SMOTE improved the overall confusion matrix reading

3.1.6. **Feature Selection:** used f_regression for feature selection, I have discussed more on this topic in the research part.

3.1.7 **Dimensionality Reduction:** Not applicable, a number of features are only 20

3.2 Models: During my project, I have built the following models

- DecisionTreeClassifier
- GaussianNB
- SVC

- KNeighborsClassifier
- RandomForestClassifier
- GradientBoostingClassifier
- LogisticRegression
- XGBClassifier

With the dataset GradientBoostingClassifier, SVC, XGBClassifier models performed the best

3.3 Hyper-parameter optimization

3.3.1 GradientBoostingClassifier: Following parameters checked

max_depth: maximum depth of the individual regression estimators
max_features: The number of features to consider when looking for the best split
min_samples_leaf: The minimum number of samples required to be at a leaf node.
min_samples_split: The minimum number of samples required to split an internal node
n_estimators: The number of estimators as selected by early stopping
learning_rate: learning rate shrinks the contribution of each tree by learning_rate.
Subsample: The fraction of samples to be used for fitting the individual base learners.
random_state: random_state is the seed used by the random number generator

3.3.2 SVC: Following parameters checked

C: The strength of the regularization is inversely proportional to C
Gamma: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'
Kernel: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.

3.3.3 XGBClassifier: Following parameters checked

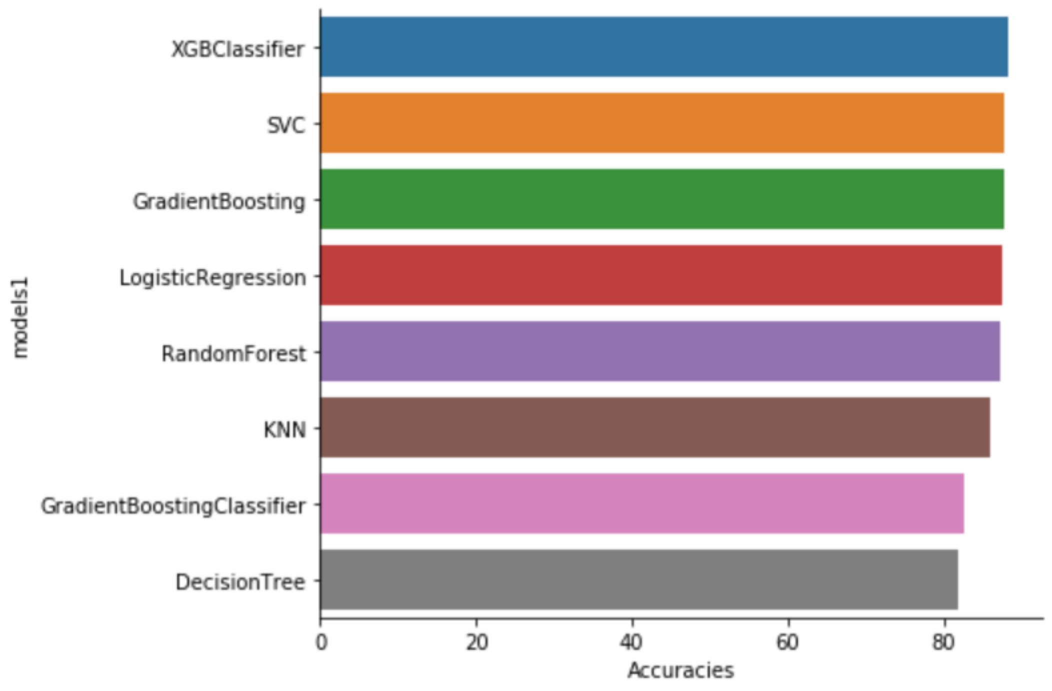
n_estimators: Number of trees to fit.
learning_rate: Boosting learning rate (xgb's "eta")
min_child_weight: Minimum sum of instance weight(hessian) needed in a child.
gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree.

subsample: Subsample ratio of the training instance.
colsample_bytree: Subsample ratio of columns when constructing each tree
max_depth: Maximum tree depth for base learners.

4. Evaluation

4.1 **Initial model exploration**

XGBClassifier	88.3
SVC	87.9
GradientBoosting	87.9
LogisticRegression	87.5
RandomForest	87.2
KNN	85.9
GradientBoostingClassifier	82.7
DecisionTree	81.8



Best Models: XGBClassifier, SVC, GradientBoosting

4.2 Hyper-parameter optimization:

Note: I have dropped rows during Hyper-parameter optimization due to computational limitations

4.2.1. XGBClassifier

Optimal values:

- colsample_bytree=0.6
- gamma=1,
- learning_rate=0.02
- max_depth=3
- min_child_weight=3
- n_estimators=300
- subsample=0.6

4.2.2. SVC

Optimal values:

- C=10
- gamma=0.01

4.2.3. **GradientBoosting**

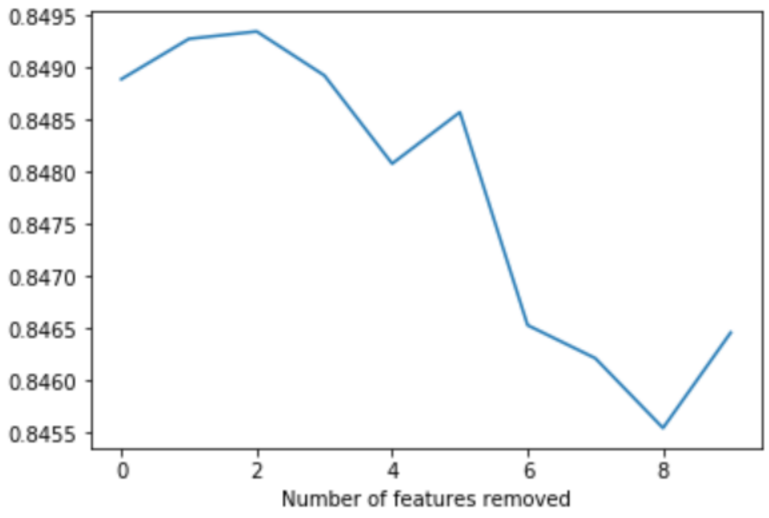
Optimal values:

- colsample_bytree=0.6
- gamma=1
- learning_rate=0.02
- max_depth=3
- min_child_weight=3
- n_estimators=300
- subsample=0.6

4.3 Feature Selection

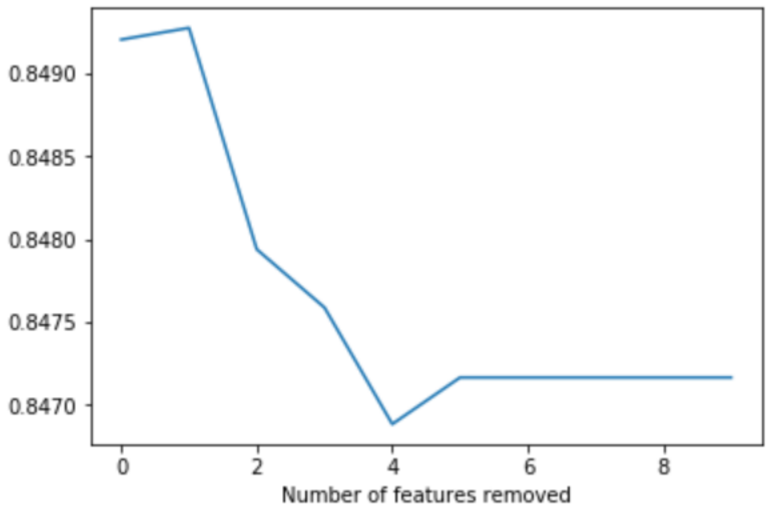
- Univariate Feature Selection
- Greedy with RFECV Feature Selection
- Chi-Square Feature Selection
- Correlations Feature Selection

4.3.1 Univariate Feature Selection



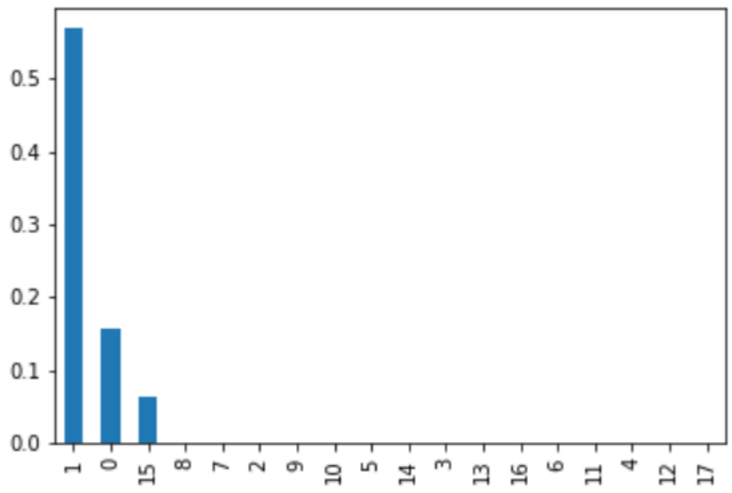
The figure above shows the accuracy plot by reduesing the number of features with least importace. We can see that our model is working better when we removed 2 least important features

4.3.2 Greedy with RFECV Feature Selection



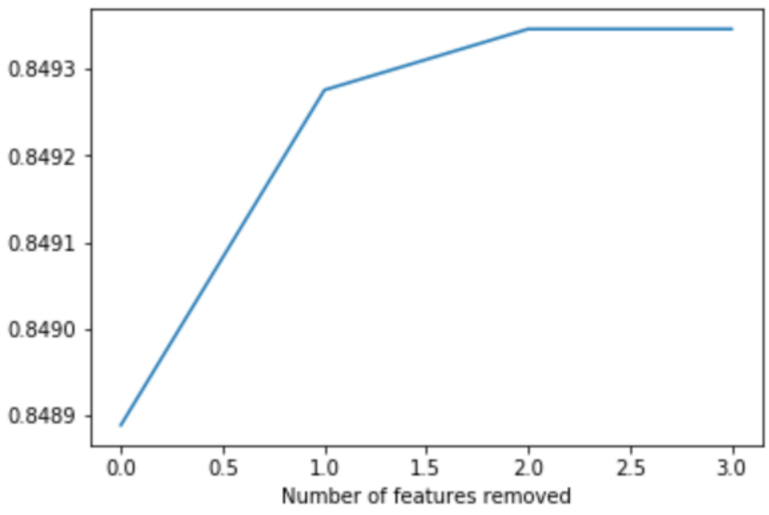
The figure above shows the accuracy plot by reduesing the number of features with least importace. We can see that our model is working better when we removed 1 least important features

4.3.3 Chi-Square Feature Selection



Index number

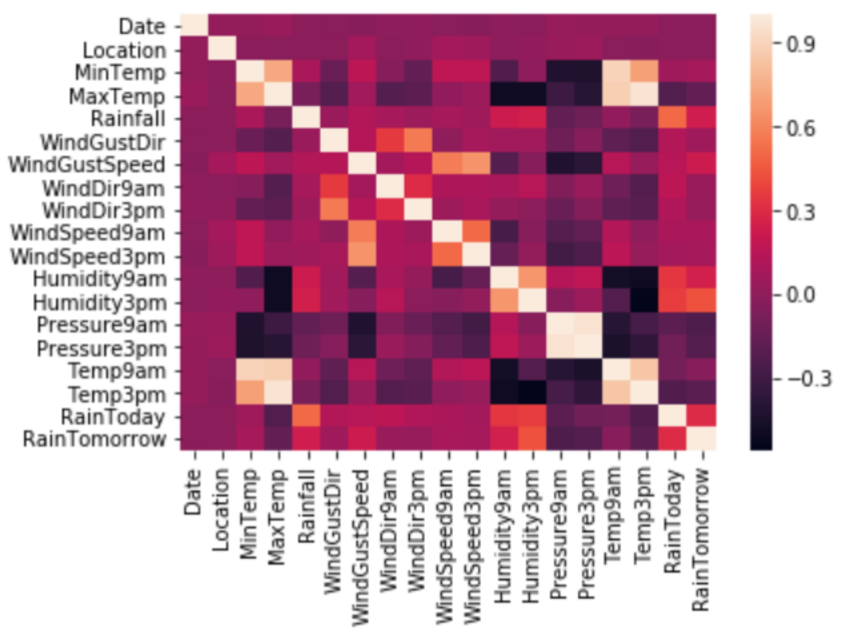
The figure above shows the independence values for the indexes, we can see that index 1 is most independent, and index 0 and 15 are independent too



With this figure we can easily see that, when we are dropping these three features. The accuracy got increased.

4.3.4 Correlations Feature Selection

```
<matplotlib.axes._subplots.AxesSubplot at 0x248c3a6efd0>
```



Heat map

Selected featues after removing the highly dependent features

```
'Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall',
'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
'Pressure9am', 'Temp9am', 'RainToday'
```

5. Conclusion

- Best model : (Before feature selection) : XGBClassifier
- Optimal parameter for XBG
  colsample_bytree=0.6
  gamma=1,
  learning_rate=0.02
  max_depth=3
  min_child_weight=3
  n_estimators=300
  subsample=0.6
- Features with approx 50% missing values : 'Evaporation','Sunshine','Cloud3pm','Cloud9am'
- Featues not important : Date, Location, Temp9am
- Best Accuracy with entire data : 84.64%

References [1]

1. https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf