Machine Learning Project Report

Elad Gashri

Faculty of Engineering, Tel Aviv University

Abstract

In this project I try to classify days whether or not they will be followed by a rainy day. This project contains classic data science techniques and supervised learning machine learning algorithms. Techniques such as dummy variables, feature standardization and PCA are performed in the preprocessing part. The algorithms k-NN, logistic regression, random forest and MLP are performed in the models part. Finally, the models are evaluated using k-fold cross-validation, ROC curves and the AUC metric.

I. Introduction

The problem of determining whether or not it will it not rain the following day is a binary classification problem.

The train dataset is composed of 22,161 days which are classified either 1: a day before a rainy day, or 0: a day before a non-rainy day. There are 25 features in the train dataset.

II. Data Exploration

There are 25 features in the dataset. 6 of them are categorical features. Further imformation about the data is presented in the Jupyter notebook of this project.

Figure 1 is the feature correlation heatmap. Although some areas with stronger correlation can be distinguished, further analysis is needed in order to determine any solid conclusions.

There are 6 categorical features in the data. Features 5, 18, 19 contain very similar values. Figure 2 is their bar plots. It seems that these features have similar distributions.

Figure 3 is the max temperature feature histogram, which presents this feature's distribution, which resembles normal distribution.

Figure 4 is the Evaporation vs Max Temperature features scatter plot. These 2 features have a correlation of 0.624, although I assumed that this correlation would be higher due to scientific connection between high temperature and evaporation.

Figure 5 is the wind gust feature box plot by the year. It shows that this feature's distribution is similar regardless of the year.

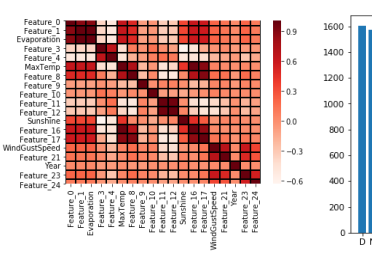

Fig 1: Features correlation heatmap

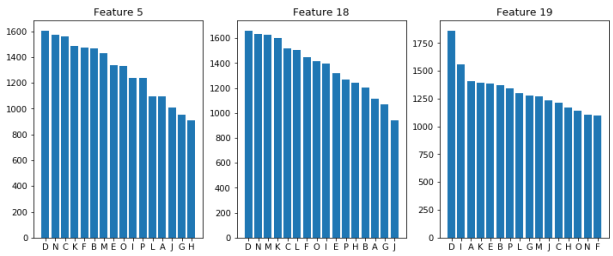

Fig. 2: Features 5, 18, 19 bar plots

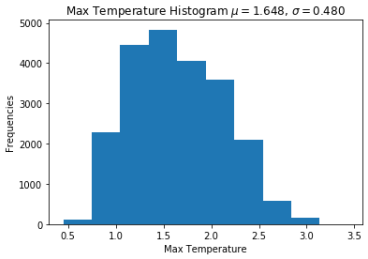

Fig. 3: Max temperature histogram

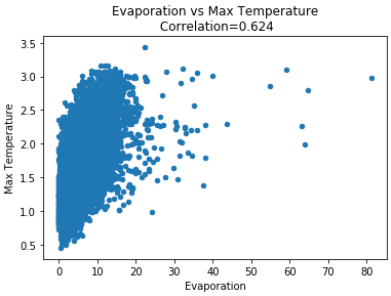

Fig. 4: Evaporation vs max temperature scatter plot

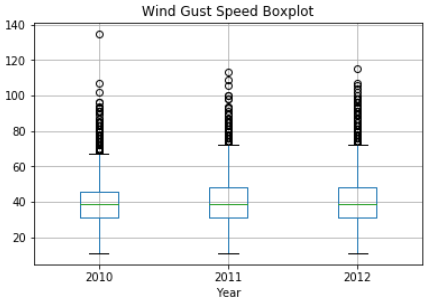

Fig. 5: Wind gust speed boxplot

III. Preprocessing

A. Feature Engineering

The engineered feature 'Hot Day' is based on the features Evaporation and Max Temperature. This is a binary feature that I believe will help determine the likelihood of it raining in the following day.

B. Outliers Removal

Figure 6 is a box plot of all the numeric features. I chose to visualize the outliers by boxplots because it shows clearly the amount of outliers each feature has. I removed outliers only from features that have shown to have a significant amount of outliers in their boxplot.

I used the interquartile range to find the outlier observations. These observations are removed from the train data.

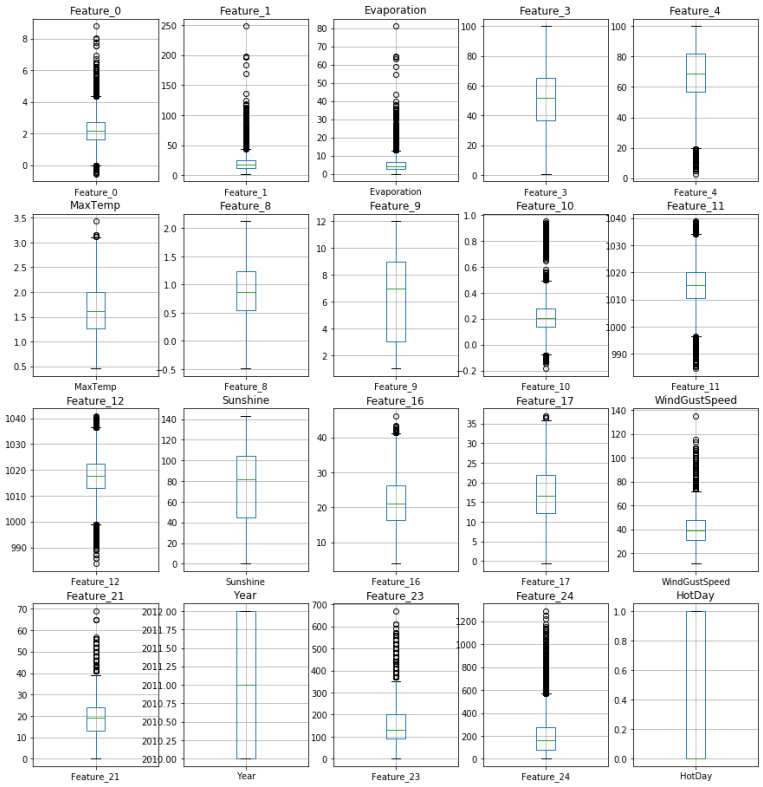

Fig. 6: Numeric features boxplots

C. Missing Values

There are 16 features that have NULL values. In the numeric features I chose to deal with those values by replacing them with either the mean or the median of the relevant feature, dependent on the way the feature distributes (figure 7). I chose the features that seem to distribute close to normal distribution. For them, I chose the mean. For the rest of the numeric features, I chose the median.

For the categorical features I replaced the missing values with the most common value.

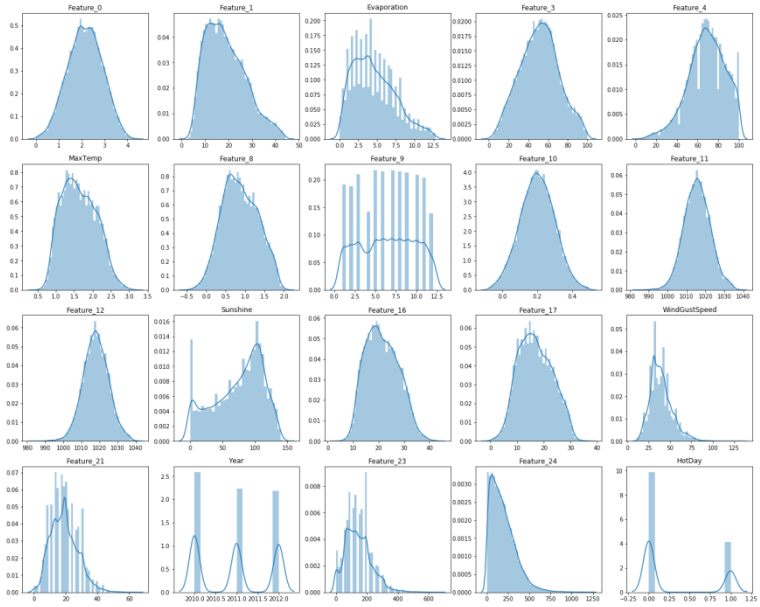

Fig. 7: Numeric features distributions

D. Categorical Features

Features 5, 18, 19 were visualized in the bar graphs in the data exploration chapter (figure 2). They were shows to have very similar values and very similar distributions. By using Cramer's V I checked their correlation. Despite what I assumed, their correlation was low. Therefore, I didn't remove any of them.

Feature 13 was modified into a binary numerical feature instead of a categorical feature. This feature's 'unknown' values were turned it into 0 or 1 on a random basis.

After examining their values and finding them numerical in nature, I also modified feature 14 and feature 6 into numerical features.

E. Data Scaling

Data scaling is crucial before starting the dimensionality reduction phase. If data scaling is not preformed, comparison between elements of different features could lead to incorrect results.

I decided to scale the features using standardization which changes the data to a standard normal distribution. Thanks to this scaling technique I will be able to find the components that maximize the variance, and to perform PCA in the dimensionality reduction phase.

F. Dummy Variables

The remaining of the categorical features were transformed into numerical featuring using dummy variables. Dummy variables are the best way to encode categorical data that is on a nominal scale.

G. Dimensionality Reduction

The dataset includes 26 features. This large number of features (high dimensionality) may cause a large test error due to overfitting. This is also known as 'curse of dimensionality'.

High dimensionality can be recognized by distinguishing that samples that are classified as the same label are distant from each other in their features values.

In high dimensionality, the amount of data that is required for generalization to take place is much larger than in lower dimensions. This causes overfitting because the model cannot generalize about samples that are distinct from each other.

I started by examining the correlations between all the features and removing the features with the least variance in any set of features that had a correlation of 0.95 or above with each other. As a result of that, 3 features were removed.

When performing PCA, I decided not to determine in advance the number of components. As shown in figure 8, I performed PCA with all possible number components: 1 to the current number of features: 23. I chose the lowest number of components that explained at least 90% of the variance of the data: 13 components.

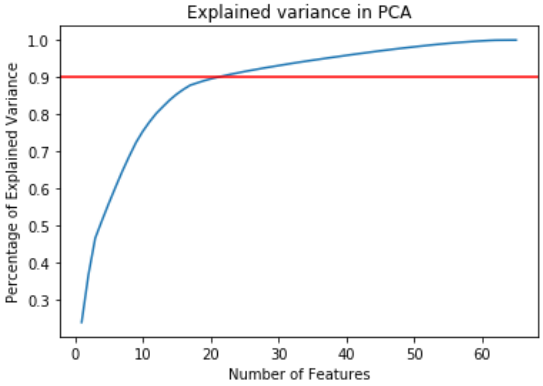

Fig. 8: Explained variance in PCA

H. Preprocessing for the Test Set

The exact same steps that were applied in the preprocessing of the train data have also been applied to the test data except for the outliers removal step.

Ⅳ. Models

A. k-Nearest Neighbors

k-NN is a simple "lazy" algorithm that classifies the sample by its closet K neighbors. I picked the value of the hyperparameter K (the number of neighbors) by the K that got the highest validation accuracy.

Figure 9 shows that at low values of K the model is slightly overfitted. If K>50 the model is not overfitted, since from that point on the train accuracy and the validation accuracy align. These results

make sense because with a smaller K, the k-NN algorithm produces a much more flexible decision boundary which can lead to overfitting.

Other hyperparameter selected without the use of validation accuracy are:

- Weights: uniform. All samples are weighted equally. There is no prior knowledge in any distinction in the importance of different samples.
- Distance metric: euclidean. Euclidean distance is the simple distance between 2 points and is the best way to find the closest samples in this case.
- Algorithm: brute force. k-NN is a simple algorithm that is best used with brute force instead of a more sophisticated method.

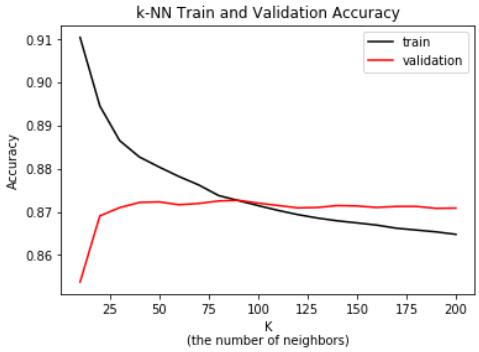

Fig. 9: k-NN train and validation accuracy

B. Logistic Regression

Logistic regression is an algorithm that gives the probability of a sample to be classified in a label. The hyperparameter C specifies the strength of the regularization. C is the inverse of the regularization parameter. A smaller C means a regularization with a higher penalty. This means the smaller the C is the simpler the model gets and is in less of a danger to be overfitted, but in higher risk of not learning from the data. C was picked as the value that got the highest validation accuracy.

Figure 10 shows that in very low values the model is completely useless and gets an accuracy of about 50%. These results makes sense: in very low values of C the regularization is very strong. Strong regularization leads to a simple model that is unable to learn from the data.

Figure 10 also shows that the model is not overfitted since there are no differences between the train accuracies and the validation accuracies.

Other hyperparameter selected without the use of validation accuracy are:

- Regularization method: lasso regression (L1). In L1 regularization feature coefficients can be zero. This would result in fewer features in the model, which would help prevent overfitting.
- Optimization algorithm: liblinear. This algorithm is a good choice when using logistic regression L1 regularized classifiers.

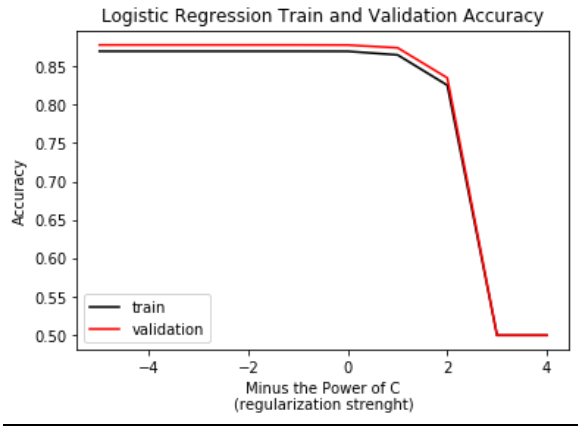

Fig. 10: Logistic regression train and validation accuracy

C. Random Forest

Random forest is an ensemble algorithm. It is made up of many decision trees. The data in each decision tree is randomly samples using boostrapping. The number of the decision trees is a hyperparameter. I picked the number of trees by the value that got the highest validation accuracy.

Figure 11 shows that although the number of trees is an important hyperparameter that is supposed to reduce the variance, it has no effect on this data.

Figure 11 also shows that the model is not overfitted since there are no significant differences between the train accuracies and the validation accuracies.

Other hyperparameter selected without the use of validation accuracy are:

- Impurity measure: gini impurity. Measures how to split a tree in each step.
- Minimum number of samples required to split an internal node: 10. A minimum number was set in order to avoid an overly complicated model and so to avoid overfitting.
- Minimum number of samples required to be at a leaf node: 100. Same as the previous hyperparamater, this help to avoid overfitting. The value in the leaf is very important and impacts the final classification, so that's why the value was set higher than the previous hyperparameter.

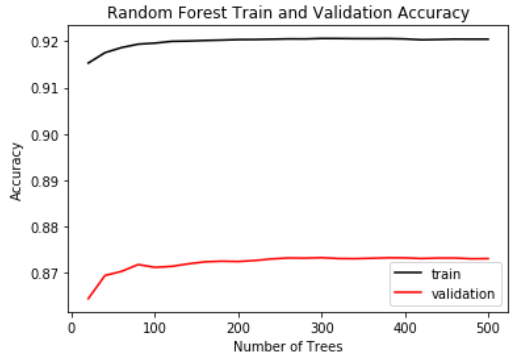

Fig. 11: Random forest train and validation accuracy

D. Multi-Layer Perceptron

A perceptron is a linear binary classifier algorithm that is based on the biological neuron. A multi-layer perceptron is an expansion of the perceptron algorithm, with at least one "hidden" layer, which allows it to perform non-linear classification.

The hyperparameters for this model were either selected in advance or selected using the function GridSearchCV that performs exhaustive search to find the best hyperparameters values.

Hyperparameters selected in advance are:

- Activation function: ReLU. Classic activation function that outputs the input if it is positive or 0 otherwise.
- Weights optimization algorithm: stochastic gradient descent. Most common machine learning optimization algorithm.

Hyperparameters selected using exhaustive search:

- Hidden layers: 1 hidden layers with 100 neurons. Too many additional hidden layers may risk the model being overfitted.
- Initial learning rate: 0.001. A small initial learning rate assures that the optimization will not get stuck at a local minimum.

Figure 12 shows the selection of the final hyperparameter alpha: the idge regularization (L2) parameter. In very high alpha values the model's performance slightly decreased. Like in the logistic regression model, strong regularization leads to a simple model that has difficulties learning from data well.

Figure 12 also shows that the model is not overfitted since there are no significant differences between the train accuracies and the validation accuracies.

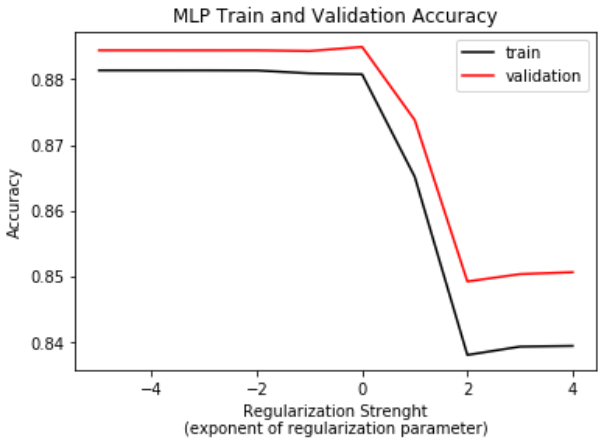

Fig. 12: MLP train and validation accuracy

Ⅴ. Model Selection and Results

For each of the models I performed k-fold cross validation (with k=10). The results are presented with ROC curves in Figures 13-16. In each graph each gray line represents the curve for a specific k-fold and the red line represents the mean curve.

I compared the models by the model by their mean AUCs. The AUCs were measures on the validation sets. The MLP model received the highest accuracy of 87.6%.

Figure 17 presents the MLP model confusion matrix. The 4 values in the confusion matrix are:

True positive (TP): a day was correctly classified as a day before a rainy day.

False positive (FP): a day was incorrectly classified as a day before a rainy day.

False negative (FN): a day was incorrectly classified as a day before a non-rainy day.

True negative (TN): a day was correctly classified as a day before a non-rainy day.

The MLP model was trained again using the train whole data (without splitting it into train and validation sets like before). The prediction results were produced to an excel document.

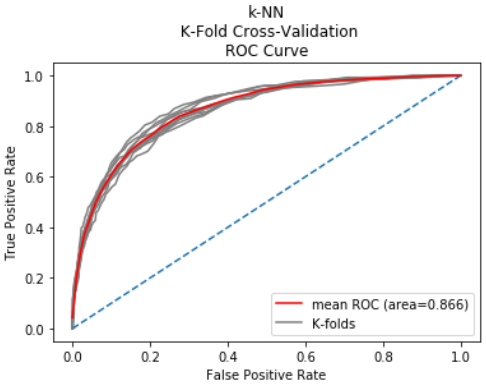

Fig. 13: k-NN k-fold cross-validation ROC curve

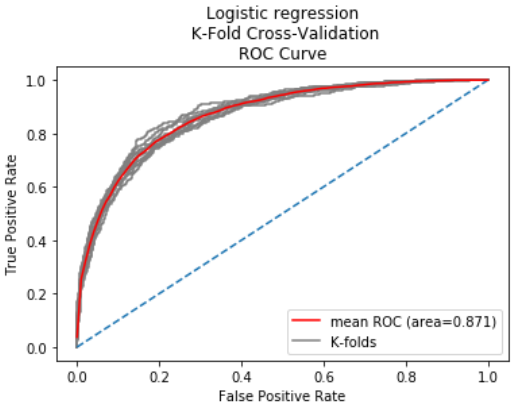

Fig. 14: Logistic regression k-fold cross-validation ROC curve

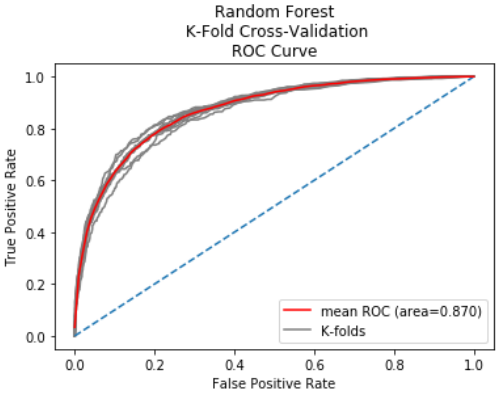

Fig. 15: Random forest k-fold cross-validation ROC curve

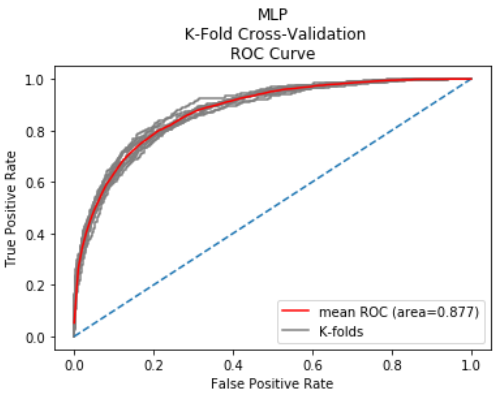

Fig. 16: MLP k-fold cross-validation ROC curve

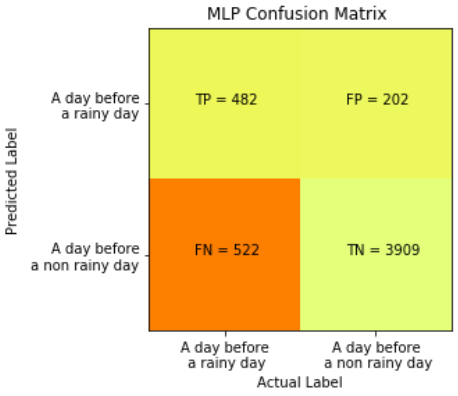

Fig. 17: MLP confusion matrix

VI. Conclusions

In all the models performed in this project, I received good results with mean accuracies around 87%. None of the models showed signs of overfitting, which may be credited to the preprocessing done to the data (especially the dimensionality reduction part).

All the models were consistent with their good accuracies, except for logistic regression in very high regularization strengths. As explained in the models part, this is because high regularization makes it difficult for the model to learn from the data.

The model that was selected to predict the test data is the MLP model. The fact that it was trained again over 100% of the train data (in contrast to 75% of the train data before) may lead to it resulting in even higher accuracies for the test data.