Name: Anjali Sharma

Email address: anjaleana@gmail.com

Contact number: 7607219812

Anydesk address:

Years of Work Experience: **3.6yrs**

Date: **14th Aug 2021**

Self Case Study -1: Don't Overfit! ||

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started:
https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

Overview

*** Write an overview of the case study that you are working on. ***(MINIMUM 200 words)*** ***

Introduction

Overfitting is one of common problem in Supervised Machine Learning especially when we have a few data points on training set, the less able is our model to generalize on unseen data or training dataset.

The process of creating a model in machine learning. The first step is to clean and visualize the data, understand the behavior of data. After that, we are splitting into training and test dataset. Let's assume 80% for training and 20% for test. We used any machine learning algorithm like **Logistic Regression, KNeighbors Classifier** etc. and

trained the model on training set. Used evaluation metrics precision, recall and accuracy. We get 90 % accuracy on training but 50% accuracy on test data(unseen data), this is called as **"Overfitting" .** Overfitting is also known as "**high - variance**". It happens many time due to unbalanced data set, few features on training data, lots of parameters in Neural Network Model.

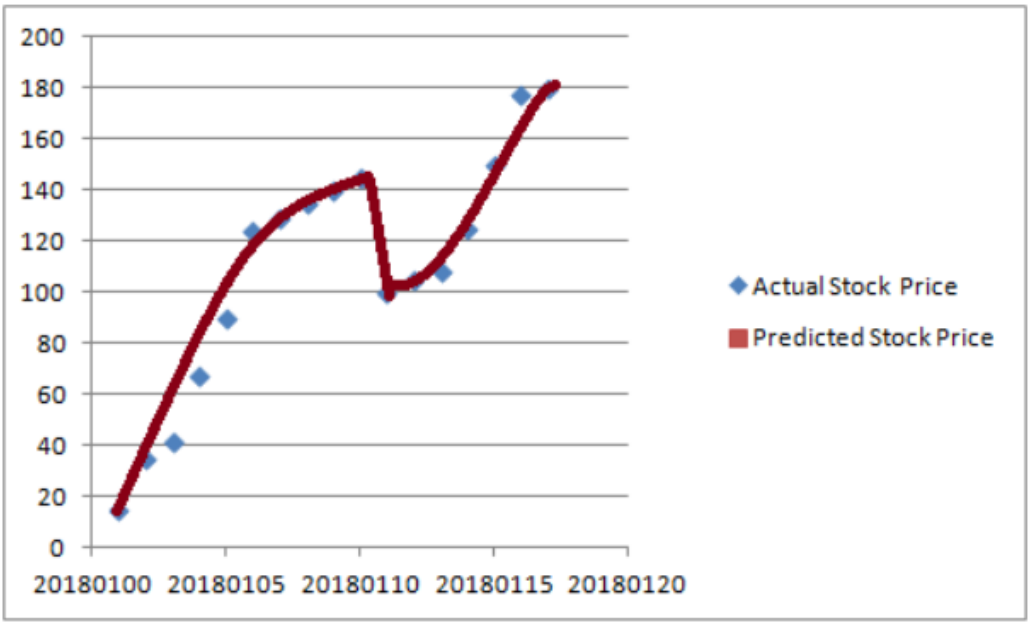
Example :

1. If some of sportsmen are good in training but bad at games. That's means it is leading overfitting problem.
2. To get good resume for Job Interview, we created the model using 10,000 sample dataset and got 90% accuracy while training the model. Later, when we used the same model to predict on unseen resume got 40 % accuracy. This is overfitting problem which is most common in real life.

We are taking Stock Market as business constraint. We want to predict the future price movements of a stock. Once data is gathered, cleaned, scaled and transformed, you split the data into training and testing datasets. Furthermore, you feed the training data into your machine learning model to train it.

Once the model is trained, you decided to test accuracy of your model by passing in test dataset.

You got below plot on training set



It looks like as it is doing great job at predicting the stock price. **However, it is known as Overfitting.** Because it has learned training data so well that it can not generalize well to make prediction on unseen data.

Source of data

To solve the overfitting problem, we are going to use kaggle dataset. Kaggle has launched a competition "Don't Overfit ||" problem. We are provided with training dataset with 250 points and test datasets with 19750 points. On training datasets, we have 250 rows and 302 columns. Out of them, first two columns indicate label and target values. Our training dataset contain the features from 0 to 299.

We have to do EDA, if we found imbalanced dataset, Hence calculating the **auc score** will be good instead of calculating the accuracy.

For performance metrics, we use the confusion matrix and calculate the F1 – score.

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. it is mandatory to write a brief description about that paper. Without understanding of the resource please don't mention it***

Research Paper url https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf

Blog1 : https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

Blog2: https://medium.com/fintechexplained/the-problem-of-overfitting-and-how-to-resolve-it-1eb9456b1dfd

Blog3 : https://towardsdatascience.com/dont-overfit-ii-how-to-avoid-overfitting-in-your-machine-learning-and-deep-learning-models-2ff903f4b36a

Blog4: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/

Blog5 : https://medium.com/analytics-vidhya/post-pruning-and-pre-pruning-in-decision-tree-561f3df73e65

https://rikunert.com/SMOTE_explained

Why

1 . Overfitting is a fundamental issue in supervised machine learning. It happens due to the presence of noise, limited size of training data as well as complexity of classifiers.

2. Because of the existence of overfitting, the model performs perfectly on training data but poorly on testing data. This is due to overfitted model has difficulty coping with pieces of information in the testing data set.

We have observed solutions for avoiding overfitting problems based on my research.

Regularization

A large number of features make the model more complex. Instead of discarding some features randomly, we can add regularization terms into the cost function that would cut down the effect of less features by minimizing their weights. The regularization parameters, known as penalty factors, are introduced which control the parameters and ensure that model is not over-training itself on the training data. These parameters are set to smaller values to eliminate the overfitting. When the coefficients take large value then regularization parameters penalize the optimization function.

L1 regularization is set to "0" for less important features. L2 regularization is set to close to "0" weights for less important features.

L1 regularization knowns as **Lasso,** it is feature selection tools and it can completely eliminate non-important features. The penalty term is absolute of magnitude of coefficient which ensures that features don't end up applying high weight on the prediction of the algorithm.

L2 regularization is known as **Rigde,** the definition of L2 penalty term is to square of magnitude of coefficients. It makes the weights which is having less important feature is set to close to zero. But not exactly Zero.

L1 creates sparsity but L2 creates the Dense matrix.

Feed More Data

We should have more relevant data so that models are trained, validated and tested. We have should have proper balanced data, while training the model, need to split 60% data to train,20% to validate and 20% for test.

Early Stopping:

This strategy is used in neural network. If we apply early stopping, it will stop learning on validation data if the accuracy of the algorithm is not increasing.

Including paper, early stopping proposing the idea to find out the exact point after this, or before this, if we won't stop learning the model, the overfitting/underfitting problem happens.

Practically, to find out the exact point, we need to keep track of accuracy on test data. We compute the accuracy at the end of each epoch and stop training when accuracy on test data stops improving.

Instead of test data, we can keep track of validation data which figure out a set of hyperparameters that can be used to evaluate the accuracy for unseen data(test data).

Feature Selection:

Feature selection is more important when number of features are very large. We can assist our algorithm by feeding in only those features that are really important. Feature selection enables machine learning algorithm to train faster, it reduces the complexity of model and makes it easier to interpret. It improves accuracy of model and reduces the overfitting.

Network-Reduction:

The complexity of model can be reduced by eliminating the irrelevant data (i.e. Noise), which in turn, would aid model to prevent overfitting and perform on unseen data, this is called as pruning, which is widely used in decision tree. There are two types of pruning, pre-pruning and post pruning.

Pre- pruning method tries to stop tree-building process early before it produces the leaves with very small samples. At each stage of tree, we are checking cross-validation error. If the error decrease significantly, we stop. This is call Early-Stopping in the learning algorithm. It reduces the overfitting problem in decision tree.

First Cut Approach

*** Explain in steps about how you want to approach this problem and the initial experiments that you want to do. ***(MINIMUM 200 words)*** ***

*** When you are doing the basic EDA and building the First Cut Approach you should not refer any blogs or papers ***

We need to read the data and understand the behavior of each feature by applying EDA technique. How each feature is followed which type of distribution, is dataset balanced or not, etc. All the features are continuous(Measured), use some basic feature engineering applied using mathematics like mean, standard deviation, trigonometric function, hyperbolic function and exponential function.

1 . The dataset is imbalanced, we have to deal it first, We will use over sampling/under sampling.

Over Sampling:

Over sampling means either we can add new Synthetic data based on similar characteristics or attributes or create duplicated data of minority class to increase number of samples as same as majority class.

Under Sampling:

In case of under sampling, we are resampling the majority class to remove the data from majority to decrease number of sample as same as minority class. *The main disadvantage is that let's assume we have 990 positive class and 10 negative class. After applying under sampling, we will get 10 sample for positive class. That's means, we are loosing the much information.*

Based on conclusion, we will go for only over sampling . **SMOTE(Synthetic Minority Oversampling Technique),** (https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/)

Second Point

Since, we understood, the dataset is imbalanced, we don't use accuracy . AUROC is the correct way to be used as evaluation metrics.

Third Point:

Dataset is not highly imbalanced, but it is decent. We don't know which works better, we will try and attempt to do with **SMOTE and without SMOTE.**

Fourth Point:

Since it is imbalanced data set, we can't lead model's prediction into **uncertainty**. We need to **calibrate** our models

Last Points:

We have to bring all data within bound range which is known as feature scaling. There are two ways to that : Normalization and standardization.

Formula of Normalization

$$\mathbf{X}_{norm} = X - \min(X) / \max(X) - \min(X)$$

Formula of Standardization

$$\mathbf{X}_{std} = X - \text{mean}(X) / \text{std}(X)$$

There is no rule of thumb, that which works better, We will try both of them.

We used 7 Model using hyper-parameters.

1. KNN(k-nearest neighbors)
2. Logistic Regression (Linear)
3. Support Vector Classifier(Non-Linear)
4. Random Forest (Non-Linear)
5. XGBoost (Non-Linear)
6. Stacking classifier
7. Decision Tree (Non - Linear)

For feature selection, we have four types

1. Filter Methods
2. Wrapper Methods
3. Embedded Methods
4. Hybrid Methods

The Wrapper methods usually result in better predictive accuracy than filter method.

We used forward feature selection and Exhaustive Feature selection.

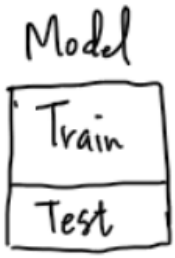
Feature Selection:

To find top 'n' importance features, First we need to train the model. At the same model, we used to find best features that can give more importance to that model. We used **forward feature selection.**

The idea is simple:

1. We need to initialize the number of top importance features of your desire.
2. Iterate over all the features in train data.
3. Stop when condition of your desire is fulfilled or cross-validation score stopped improving after adding new features.

$F_1, F_2, F_3, F_4 \ldots, F_n$



1st iteration

Each feature pass to this model & evaluate test score

$F_1 \rightarrow$ Model $\rightarrow$ Test Score$_1$

$F_2 \rightarrow$ Model $\rightarrow$ Test Score$_2$

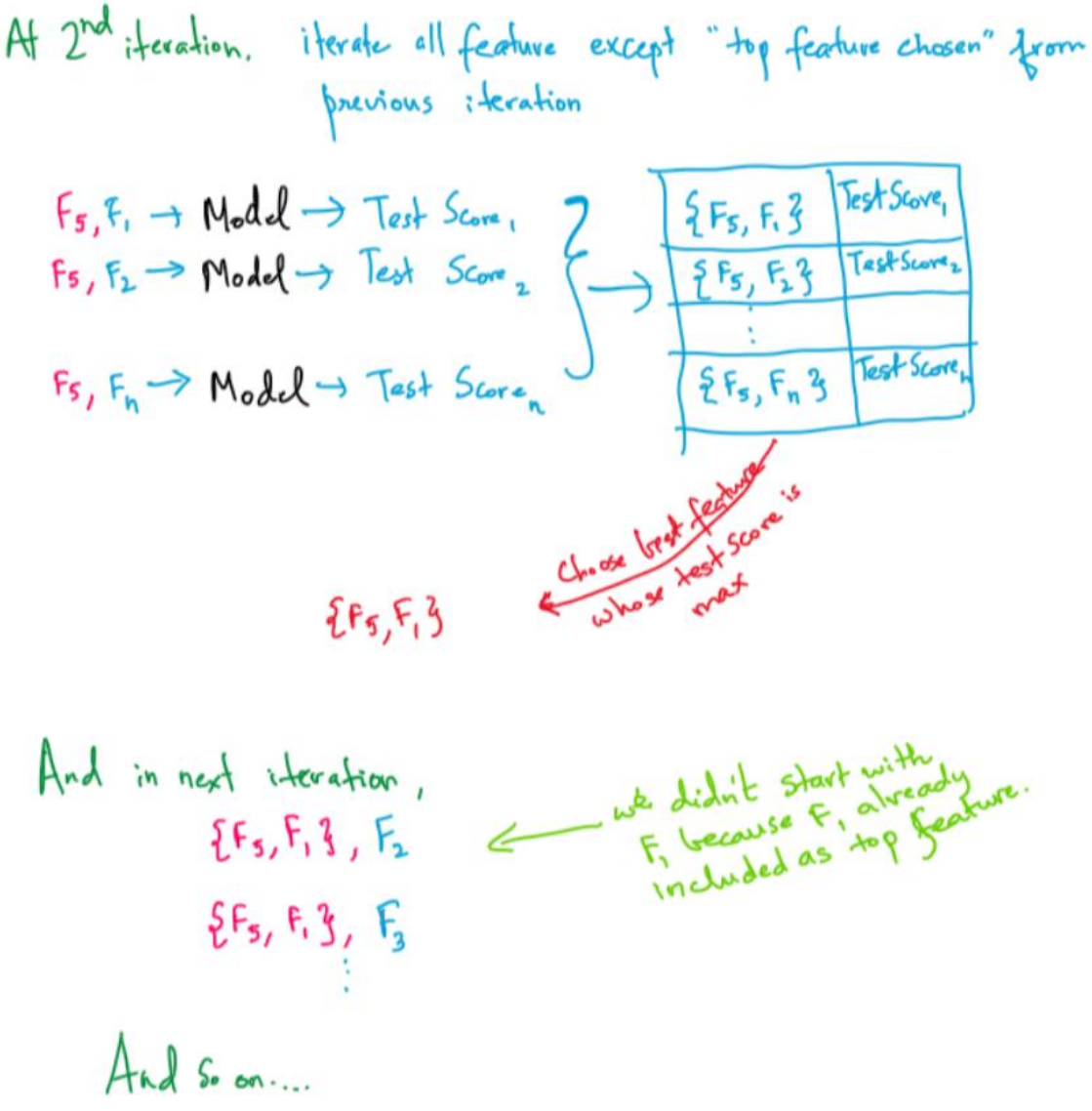
$F_n \rightarrow$ Model $\rightarrow$ Test Score$_n$

F_1	Test Score$_1$
F_2	Test Score$_2$
F_n	Test Score$_n$

choose F_x where Test Score$_x$ is max

F_5

(let say x=5)



Exhaustive Feature Selection

This is the most robust feature selection method . This is a brute-force evaluation of each feature subset. This means that it tries every possible combination of the variables and returns the best performing subset.

Embedded methods are good than filter and wrapper method. We use L1 regularization called as **Lasso regression** which penalize the coefficient by adding absolute value of magnitude of coefficient. This is also known as **L1 norm.**

We used all above mentioned technique and trained the model, validate and test model and find which is giving good Kaggle score.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument "X" (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):
        preprocess data i.e data cleaning, filling missing values etc
        compute features based on this X
        use pre trained model
        return predicted outputs
    final([time, location])
    d. in the instructions, we have mentioned two functions one with original values and one without it
    e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)
4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data

5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models