Name: Naveen Kumar Mohan

Email address: naveen97mnk@gmail.com

Contact number: 919442867607

Anydesk address:

Years of Work Experience: **2.4**

Date: **21th Nov 2021**

Self Case Study -1: Don’t Overfit - Kaggle Competition

“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)*** ***

This Case Study is based on one of the Kaggle competition problems named “Don’t Overfit!!”. It emphasizes the most necessary step in the building of models, that is to ensure that the model does not overfit the training data and leads to good predictions on new unseen data.

Business Problem:

The problem is to find a model that generalizes well, seems simple but the catch here is that we have to find it from the extremely small training samples and should work on the unseen testing samples without overfitting. The main objective of this problem is to use the existing research, algorithms, techniques or strategies that can be used to prevent the model from overfitting to the training dataset. These techniques or strategies will be used in the area of Feature Engineering, Feature Selection, Oversampling dataset.

The purpose of this case study is to solve the given problem by using the leverage that the feature engineering and feature selection will bring to the model and to show how a well-engineered feature will allow us to use a less complex model and thereby avoid overfit.

1.	**Problem type**	:	Binary Classification
2.	**Performance Metric**	:	AUCROC (Kaggle will evaluate using AUCROC)
3.	**No. of Train data point**	:	250 Data Points
4.	**No. of Test data point**	:	19,750 Data Points
5.	**No. of Features**	:	300 Features

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***

1. **Kaggle Competition-Don’t Overfit II** (https://medium.com/analytics-vidhya/kaggle-competition-dont-overfit-ii-74cf2d9deed5)
   1.1. Performed Feature Engineering techniques such as basics statistics, trigonometric function, hyperbolic function and exponential function.
   1.2. Experimented with SMOTE to oversample the minority data set.
   1.3. Experimented by applying PCA, truncated SVD to reduce the Dimensionality of the data set but that did not improve the AUC score of the dataset.
   1.4. Applied forward feature selection and improved the AUC score.
   1.5. Experimented using KNN, Logistic Regression , SVC, Random Forest, XGBoost, Stacking Classifier , Voting Classifier.
   1.6. Achieved 0.833 Private score using Voting Classifier and using only the top features from Forward Feature selection method.

2. **Don’t Overfit** (https://medium.com/analytics-vidhya/dont-overfit-645e3957f30b)
   2.1. Used probability distribution of each class of all the features and selected the features based on the observation from the distributions.
   2.2. Used Pearson Correlation coefficient, Recursive Feature Elimination and used Feature importance from Decision Tree, LASSO to select the most important features.
   2.3. Used simple Machine learning models such as KNeighborsClassifier and Logistic Regression.

    2.4. Achieved AUROC score of 0.844 using Logistic Regression with L1 Regularization.

3. **Don't Overfit! II — How to avoid Overfitting in your Machine Learning and Deep Learning Models**
(https://towardsdatascience.com/dont-overfit-ii-how-to-avoid-overfitting-in-your-machine-learning-and-deep-learning-models-2ff903f4b36a)
    3.1. Experimented with Quartile based binning during Feature Engineering.
    3.2. Experimented using SMOTE, Random Under Sampler to oversample and under sample the data points and to balance the data set.
    3.3. Experimented with Sequential Forward Selection, Sequential Forward Floating Selection, Recursive Feature Elimination.
    3.4. Experimented Using KNN, Logistic Regression, SVC, Naïve Bayes, Decision Tree, Linear Discriminant Analysis, Random Forest, XGBoost, LassoCv, LassoLarsCV, and RidgeCV.
    3.5. Experimented Using Deep learning models such as 2 layered Deep learning model and using Gaussian Dense layer to reduce the overfitting.
    3.6. Achieved AUROC private score of 0.838 using Logistic Regression and RFE.

4. **Oversampling with SMOTE and ADASYN**
(https://www.kaggle.com/residentmario/oversampling-with-smote-and-adasyn)

In this kernel the author talks about the **SMOTE**(Synthetic Minority Oversampling Technique) , the **variations of the SMOTE** and **ADASYN**(Adaptive Synthetic Sampling) methods for oversampling the data points and the characteristics of the generated artificial data points. We can use these methods to balance the dataset and improve the performance of our model.

5. **Boost Model Accuracy of Imbalanced COVID-19 Mortality Prediction Using GAN-based Oversampling Technique** (https://www.analyticsvidhya.com/blog/2020/10/lets-improve-the-model-accuracy-of-imbalanced-covid-19-mortality-prediction-using-generative-adversarial-networks-gan-based-oversampling-technique/)

   In this article they were able to increase the performance of the model by balancing the real word skewed Covid-19 data in predicting the risk of mortality through **Generative Adversarial Networks (GAN)** based oversampling technique. The Covid-19 data used in studies were trained using 222 patient records with 13 features. The data is biased as 159(72%) cases belong to the class '0′ or ‘recovered’, which is similar to the data size and the imbalance ratio we have. The GAN allows us to sample artificial points that are similar in pattern with the original data points.

6. **Loan Default Prediction - Imperial College London - Kaggle Problem 2nd Place Solution** (https://github.com/freedomljc/Loan_Default_Prediction)

   In this the author, due to lack of feature description, used operations like +-*/ between all possible two features and (a - b) * c among all possible three features and selected only the combinations of features that resulted in high Pearson correlation with the target variable. Since we also have features without any description we can also try out different brute force techniques like above and find the highly correlated combinations to improve our model during Feature Engineering.

7. **Microsoft Malware Detection - log loss of 0.0070** (https://www.kaggle.com/paulrohan2020/microsoft-malware-detection-log-loss-of-0-0070)

   In this kernel the author converted the asm file into an image representation and was able to improve the log loss by using the first 800 pixels from the image. We can also try to create an image representation of the 300 features we have and see if that will improve the performance of our model.

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 ***

Since the train data is so low and it has more features, I am planning to try increasing the size of the train dataset and reduce the number of features and try out various feature engineering. We can use the techniques that were previously employed to solve similar problems and see if that will improve our model.

Planning to try out the below mentioned technique and modeling using **Logistic Regression, Naïve Bayes etc**.... on the data and choose the combination with the best score on the test data.

- **Exploratory Data Analysis:**
    - Plot the Pair plot, PDF, CDF, Box Plot, Violin Plot of all the features and try to find any insights from them.
- **Feature Engineering:**
    - Trying out by applying the Trigonometric functions, Hyperbolic functions, Exponential on each of the features.
    - Trying out by applying operation such as +-/* between all possible combinations of two features.
    - Try to convert the data point into an image representation of the 300 features we have and use it in the model.
- **Feature Selection:**
    - Applying chi square on the features and finding the feature importance.
    - Finding correlation among the features and keeping only the highly correlated features.
    - Using model-based Feature selections.
    - Try to combine results found from all the above methods using Arithmetic mean, Harmonic Mean, etc... And finding a single score for a feature.
- **Oversampling Data Points:**
    - Removing outliers using Local Outlier Factor and applying different variations of SMOTE, ADASYN.
    - Using ADASYN, SMOTE and different variation of SMOTE
    - Using Generative adversarial networks(GAN) to generate more data points that are similar to the original data points.
- **Dimensionality Reduction:**

- Using PCA to reduce the dimensions.
- Using Autoencoders to find the low Dimensional Representation of the data.

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