CUNY MSDS – DATA622

Homework 3

**Introduction**

For this assignment, I will aim to compare performance of the Support Vector Machine (SVM) and Decision Tree (DT) model frameworks. Note that the DT frameworks include ensemble approaches like Random Forest (RF) and Gradient Boosted Trees (GBT). To compare these algorithms, I will again use the dataset of simulated online payments from Homework #2, labelled to identify fraudulent transactions. I sourced the data from Kaggle [1], but the data ultimately comes from a 2016 paper from Lopez-Rojas et al [2] that introduced a payment simulator named PaySim.

Homework 2 contains detailed exploratory data analysis (EDA), but below is a brief overview of the dataset, which provides ~6.35 million observations and includes nine features and one label:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

The target variable (“isFraud”) is a binary label indicating whether a transaction is fraudulent (1) or not (0). The dataset is heavily imbalanced, with over 99% of the transactions being non-fraudulent.

The overall approach for the assignment is as follows:

* Literature Review
* SVM Training Overview
* Performance Comparison
* Conclusions

Code for all the following analyses is saved here: https://github.com/kac624/cuny/tree/main/D622.

**Literature Review**

I will cover six articles, including the two provided by the assignment prompt. These articles cover the application of SVMs and DTs across three subject areas: COVID diagnosis, stock price prediction, and bankruptcy prediction.

* Decision Tree Ensembles to Predict Coronavirus Disease 2019 Infection [3]
* A novel approach to predict COVID-19 using support vector machine [4]
* Predicting the direction of stock market prices using random forest [5]
* Forecasting stock market movement direction with support vector machine [6]
* Bankruptcy prediction using support vector machine [7]
* Personal bankruptcy prediction using decision tree model [8]

First, I will cover the two articles related to COVID diagnosis. Article [3] — Decision Tree Ensembles to Predict Coronavirus Disease 2019 Infection — analyzes various ensemble methods like Random Forest and Boosting. It highlights the adaptability of DT models to handle imbalanced datasets, a common challenge in medical data analysis. The article finds that techniques like Balanced Random Forest and SMOTE-enhanced ensembles can significantly improve predictive performance in terms of AUROC and AUPRC metrics when compared to single tree models. Article [4] — A novel approach to predict COVID-19 using support vector machine — finds that SVM achieved accuracy of 87% in classifying patient conditions as no infection, mild infection and serious infection. The paper also highlights the importance of data visualization to support prediction.

Regarding stock market prediction, Article [5] — Predicting the direction of stock market prices using random forest — employs technical indicators derived from exponentially smoothed time series data as input features for the model. These indicators help predict future stock price behavior by identifying trends such as overbought or oversold conditions. The study emphasizes the model's ability to handle the complex and noisy nature of stock market data effectively. However, it suffers from high computational requirements due to its complexity. Article [6] — Forecasting stock market movement direction with support vector machine —focuses on SVM's capability to capture patterns in data that are non-linear and complex, which is often the case in financial time series. The performance of SVM is analyzed against other models like ARIMA, displaying its superior capability in handling noisy, non-stationary financial data​.

Finally, there are two articles covering these models’ application for bankruptcy prediction. Article [7] — Bankruptcy prediction using support vector machine — focuses on optimization of kernel function parameters through a rigorous grid-search method. The approach enhances SVM's prediction accuracy and stability, ultimately providing solid performance in forecasting bankruptcy risk in businesses. The paper highlights the critical role of parameter selection in SVM performance, ensuring that the model adapts well to various datasets used in the study. Article [8] — Personal bankruptcy prediction using decision tree model — also leverages minority class oversampling to address class imbalance. The model ultimately yields >80% accuracy, and authors highlight the explainability of the single DT model as a meaningful advantage for this use case.

In summary, both models demonstrate strong potential for predictions on complex, non-linear data. The DT approaches seem to show more promise with heavily imbalanced data, and SVM approaches seem especially reliant on careful calibration of kernel hyperparameters. Moreover, both models appear to carry significant computational expense.

**Model Training**

To compare SVM and DT models myself, I trained an SVM model (using the scikit-learn implementation) on the dataset from Homework 2. I started with a baseline model using all default hyperparameters. As with Homework 2, I used an 80%-10%-10% split between training, validation, and testing subsets. Below is a summary of the baseline results (i.e., using default hyperparameters) across training and validation subsets, along with the baseline results for an RF model, which was the best performing model from Homework 2 and will serve as the benchmark for SVM.

*Figure 1: Baseline Model Results*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | SVM | RF |
| **Train** | Accuracy | 1.00 | 1.00 |
| Precision | 0.99 | 1.00 |
| Recall | 0.46 | 1.00 |
| F-1 Score | 0.63 | 1.00 |
| **Validation** | Accuracy | 1.00 | 1.00 |
| Precision | 1.00 | 0.96 |
| Recall | 0.46 | 0.79 |
| F-1 Score | 0.63 | 0.86 |

Overall, the initial performance of SVM is significantly worse than RF. First, it is important to note that, due to the heavy imbalance of our data, accuracy is not a helpful measure of performance. Second, the precision is very high, indicating that the SVM model produces very few false positives. By contrast, however, recall is quite low, indicating that the model produces many false negatives. In addition to poor baseline performance, this initial training highlighted another drawback of SVM: computational expense. Whereas the RF model took only 1.5 minutes to train, the SVM model requires 160 minutes (more than 100x!).

The literature on SVM consistently highlights the importance of optimizing kernel parameters, so I worked to tune the SVM model’s hyperparameters. The various hyperparameter values used in tuning are shown in the code snippet below.

```

hyperparams = {

    'svm': {

        'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],

        'C': [0.1, 1, 10, 100],

        'gamma': ['scale', 'auto', 0.1, 1, 10],

        'ceof0': [0, 0.5, 1]

    }

}

```

However, because of the very long training time mentioned above, I had to make two compromises. First, I had to run the tuning protocol on a small subset of the data (just 5% of the ~6.35 million observations). Second, I could not run a full grid search that considered all combinations of potential hyperparameter values. Instead, I sequentially ran a series of separate tuning scripts, iterating over values of one hyperparameter while keeping all the others fixed. Because of these compromises, my final set of hyperparameters (shown below) is likely sub-optimal.

```

hyperparams = {

    'svm': {

'kernel': 'rbf',

'C': 100,

'gamma': 1,

'coef0': 0

}

}

```

With these hyperparameters, I re-trained the models one final time. This time, however, I used both the combined training and validation subsets for model fitting, then evaluated performance with the holdout test subset. Figure 2 summarizes the results, along with results from the optimized RF model from Homework 2, for comparison.

*Figure 2: Final Model Performance on Holdout Test Data*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | SVM | RF |
| **Train + Valid** | Accuracy | 1.00 | 1.00 |
| Precision | 0.98 | 1.00 |
| Recall | 0.69 | 1.00 |
| F-1 Score | 0.81 | 1.00 |
| **Test** | Accuracy | 1.00 | 1.00 |
| Precision | 0.98 | 0.96 |
| Recall | 0.69 | 0.90 |
| F-1 Score | 0.81 | 0.93 |

As indicated by the literature, the updated hyperparameters appeared to drive significantly better performance for the SVM model. Precision remained near 1, and recall increased from 0.46 to 0.69, driving up the F-1 score from 0.63 to 0.81. However, performance still appears better with RF.

**Conclusions**

This assignment highlights several important conclusions.

* Both SVM and DT frameworks appear well suited to handle complex data with potentially non-linear relationships.
* Computational efficiency can be an impediment for both model frameworks, but the problem appears much more pronounced with SVM models.
* SVM models are especially responsive to hyperparameter tuning, especially regarding kernel choice. However, long training times can make grid searches computationally expensive and hinder the search for optimal hyperparameters.
* Overall, RF appears to outperform SVM on this data. SVM does offer higher precision, but at the cost of significantly lower recall. Depending on the use case, this condition may be desirable. In the case of fraud detection, however, high recall is typically important, so RF appears to be the better choice for this data.

**Sources**

[1] Rupak Roy (2022). "Online Payments Fraud Detection Dataset." Sourced from https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset/discussion/330548.

[2] Edger Alonso Lopez-Rojas; Stefan Axelsson; & Ahmed Elmir (2017). “PAYSIM: A financial mobile money simulator for fraud detection.” Sourced from https://www.researchgate.net/publication/313138956\_PAYSIM\_A\_FINANCIAL\_MOBILE\_MONEY\_SIMULATOR\_FOR\_FRAUD\_DETECTION.

[3] Soham Guhathakurata; Souvik Kundu; Arpita Chakraborty; & Jyoti Sekhar Banerjee (2021). “A novel approach to predict COVID-19 using support vector machine.” Sourced from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8137961/.

[4] Amir Ahmad; Ourooj Safi; Sharaf Malebary; Sami Alesawi; & Entisar Alkayal (2021). “Decision Tree Ensembles to Predict Coronavirus Disease 2019 Infection: A Comparative Study.” Sourced from https://www.hindawi.com/journals/complexity/2021/5550344/.

[5] Kaiming He; Xiangyu Zhang; Shaoqing Ren; & Jian Sun (2016). “Deep Residual Learning for Image Recognition.” Sourced from https://arxiv.org/pdf/1605.00003.

[6] Wei Huanga; Yoshiteru Nakamoria; & Shou-Yang Wang. (2005). “Forecasting stock market movement direction with support vector machine.” Sourced from https://svms.org/finance/HuangNakamoriWang2005.pdf.

[7] Jae Min & Young-Chan Lee (2005). “Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters.” Sourced from https://www.researchgate.net/publication/222580945\_Bankruptcy\_prediction\_using\_support\_vector\_machine\_with\_optimal\_choice\_of\_kernel\_function\_parameters.

[8] Sharifah Heryati Syed Nor; Shafinar Ismail; & Yap Bee Wah (2019). “Personal bankruptcy prediction using decision tree model.” Sourced from https://doi.org/10.1108/JEFAS-08-2018-0076.