P33 Global Data Lab Weekly Report

Kipkemoi Vincent

July 12, 2024

Introduction

Anomaly detection in financial markets plays a crucial role in identifying irregularities or outliers that deviate significantly from normal patterns, thus offering invaluable insights for risk management and fraud detection. As financial transactions grow increasingly complex and voluminous, traditional methods often struggle to cope with the scale and diversity of data generated. This is where advanced techniques such as PyOD (Python Outlier Detection) and Microsoft AutoML (FLAML) come into play. PyOD provides a comprehensive suite of algorithms tailored for anomaly detection, including statistical approaches, proximity-based methods, and machine learning models like Isolation Forest and Autoencoders. These algorithms excel in detecting anomalies across various financial metrics such as transaction amounts, frequency, and temporal patterns, offering flexibility and robust performance in differentiating legitimate transactions from potentially fraudulent ones.

Microsoft AutoML, powered by FLAML (Fast Lightweight AutoML), represents a significant leap forward in automating the machine learning pipeline for anomaly detection in financial markets. FLAML optimizes hyperparameters and selects the best model from a range of candidates, enabling efficient and accurate anomaly detection without requiring extensive manual intervention. By harnessing the power of automated machine learning, financial institutions can expedite the deployment of anomaly detection systems, enhance detection accuracy, and adapt rapidly to evolving financial landscapes. This capability not only strengthens security measures against fraud but also enhances decision-making processes by providing timely insights into emerging market trends and anomalies that could impact financial stability. Thus, integrating PyOD and Microsoft AutoML (FLAML) empowers financial institutions to stay ahead in detecting anomalies, mitigating risks, and safeguarding financial assets with heightened precision and efficiency.

Objective

In this work, I have considered an imbalanced dataset of credit card frauds (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud) with the target labels being authentic and fraudulent and evaluated a number of PyOD and AutoML (FLAML) models in identifying transactions that are, in some sense, different from the usual, authentic transactions. As the data is highly imbalanced, I have implemented different balancing approaches such as undersampling, oversampling and SMOTE, and use the best model to evaluate the performance under each case. Furthermore, As oppose to using 'accuracy' alone as evaluation metric, I have used different evaluation metrics such as Precision, Recall, F1-score and ROC-AUC considered robust especially when dealing with imbalanced dataset.

Data

The dataset contains information on the transactions made using credit cards by European cardholders, in two particular days of September 2013 . It presents a total of 284807 transactions, of which 492 were fraudulent. Clearly, the dataset is highly imbalanced, the positive class (fraudulent transactions) accounting for only 0.173% of all transactions. The columns in the dataset are as follows:

Time: The time (in seconds) elapsed between the transaction and the very first transaction

V1 to V28: Obtained from principle component analysis (PCA) transformation on original features that are not available due to confidentiality

Amount: The amount of the transaction

Class: The status of the transaction with respect to authenticity. The class of an authentic (resp. fraudulent) transaction is taken to be 0

Evaluation Metrics

Let TP, TN, FP and FN respectively denote the number of true positives, true negatives, false positives and false negatives among the predictions made by a particular classification model. Below we give the definitions of some evaluation metrics based on these four quantities.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of total predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive predictions}} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive cases}} = \frac{TP}{TP + FN}$$

$$F_1\text{-Score} = \text{Harmonic mean of Precision and Recall} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F_2\text{-score} = \frac{5 \times TP}{5 \times TP + 4 \times FN + FP}.$$

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is a metric used to evaluate the performance of binary classification models. It measures the ability of the model to distinguish between classes by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various thresholds.

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is given by:

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(FPR^{-1}(t))\, dt$$

where:

- $\text{TPR}(t) = \frac{\text{TP}(t)}{\text{P}}$ is the True Positive Rate at threshold t,
- $\text{FPR}(t) = \frac{\text{FP}(t)}{\text{N}}$ is the False Positive Rate at threshold t,
- $\text{TP}(t)$ is the number of true positives at threshold t,
- $\text{FP}(t)$ is the number of false positives at threshold t,
- P is the total number of positive instances,
- N is the total number of negative instances.

Feature Selection

High dimensionality creates difficulties for anomaly detection because, when the number of attributes or features increase, the amount of data needed to generalize accurately also grows, resulting in data sparsity in which data points are more scattered and isolated. This data sparsity is due to unnecessary variables, or the high noise level of multiple irrelevant attributes, that conceal the true anomalies. This issue is widely acknowledged as the **curse of dimensionality**.

In the problem at hand, we have 30 features. We aim to keep only those which help substantially in discriminating between authentic and fraudulent transactions. More specifically, we compare the distribution of each feature for both the target classes. If a feature has similar distributions for both authentic and fraudulent transactions, then it is not likely to contribute much in the process of classifying a transaction as **authentic** or **fraudulent**. However, if a feature has very different distributions for different target classes, then it plays a far more significant role in the same process. We plot the distributions and select the features exhibiting fairly distinct distributions across the target classes. Fig. 1 shows features which exhibited very different distributions for different target classes and hence were selected for this analysis

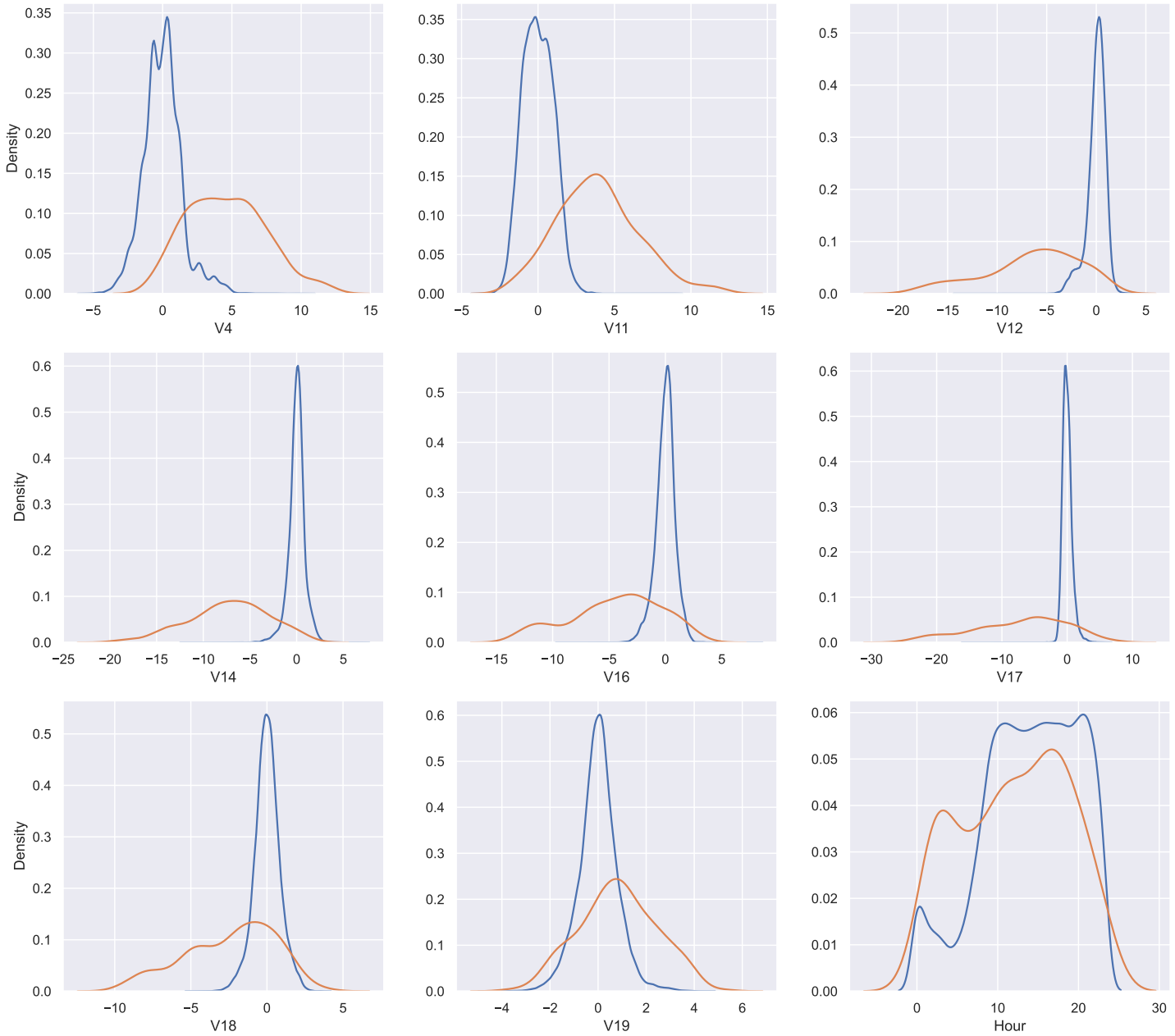


Figure 1: Features exhibiting very different distributions for different target classes

Results and Discussion

In this work, I have evaluated a number of anomaly detection models using credit card data. Based on the F1-score, recall, precision, accuracy, and ROC AUC scores provided in Fig 2. for different anomaly detection algorithms, FLAML's ExtraTreeClassifier emerges as the standout performer, showcasing exceptional results across multiple metrics. With an impressive F1-score of 0.744, FLAML's ExtraTreeClassifier demonstrates a robust balance between precision and recall, indicating its ability to accurately identify anomalies while minimizing false positives and negatives. This performance is underpinned by its high recall score of 0.684, signifying its capacity to detect a significant proportion of true anomalies within the dataset. Additionally, FLAML's ExtraTreeClassifier achieves the highest precision score of 0.817, underscoring its reliability in flagging anomalies with minimal misclassifications.

In terms of overall classification accuracy, FLAML's ExtraTreeClassifier achieves near-perfection at 0.999, reflecting its capability to correctly classify the vast majority of instances. This exceptional accuracy is mirrored in its ROC AUC score of 0.966, indicating strong discriminatory power in distinguishing between normal and anomalous data points. Conversely, models such as Local Outlier Factor (LOF) and One-Class SVM (OCSVM) exhibit notable weaknesses in certain metrics. LOF, despite achieving a respectable accuracy of 0.989, suffers from a low F1-score and recall of 0.0, suggesting challenges in correctly identifying anomalies, particularly in imbalanced datasets where anomalies are rare. Similarly, OCSVM shows a

precision of 0.015, indicating a higher rate of false positives compared to FLAML's ExtraTreeClassifier and HBOS.

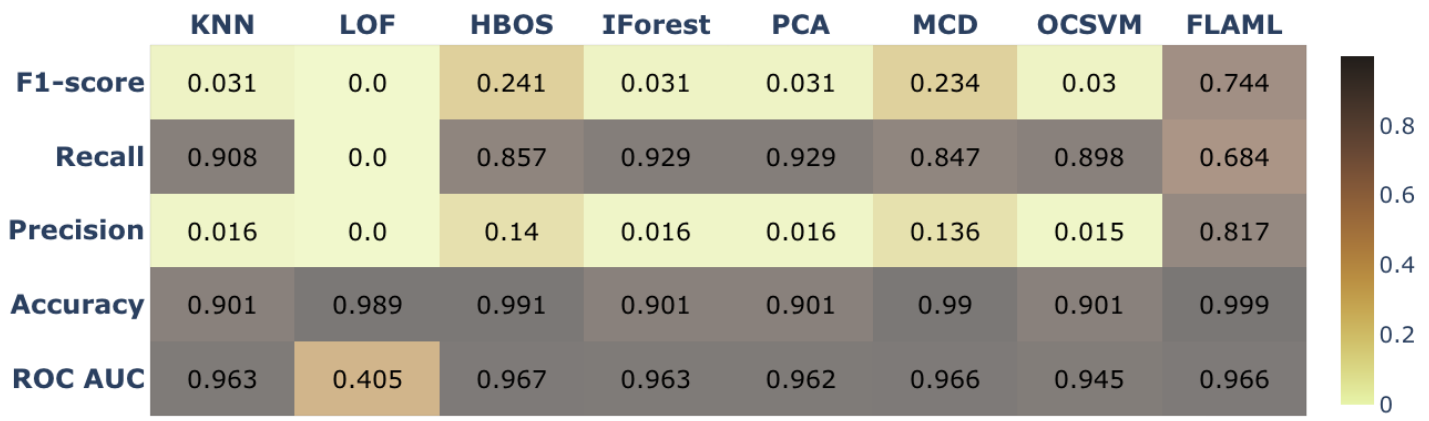


Figure 2: Performance evaluation of different models based on F1-score, recall, precision, accuracy, and ROC AUC scores

Histogram-based Outlier Score (HBOS) also demonstrates commendable performance, particularly with its high F1-score of 0.241 and robust ROC AUC score of 0.967. HBOS strikes a balance between precision (0.14) and recall (0.857), making it effective in identifying anomalies with fewer misclassifications. Models like Isolation Forest (IForest) and Minimum Covariance Determinant (MCD) exhibit strong recall scores above 0.8, indicating their effectiveness in capturing true anomalies, albeit with slightly lower precision compared to FLAML's ExtraTreeClassifier and HBOS.

In conclusion, FLAML's ExtraTreeClassifier stands out as the top performer across various metrics, including F1-score, recall, precision, accuracy, and ROC AUC. The choice of model should be guided by specific application requirements, such as the need for high precision or recall, computational efficiency, and the nature of the dataset. Leveraging these insights enables stakeholders to make informed decisions to enhance anomaly detection strategies tailored to their operational needs and performance expectations.

This refined analysis maintains focus on FLAML's ExtraTreeClassifier throughout, providing a deep dive into its strengths and comparative advantages over other models in anomaly detection tasks.

Having found that the FLAML's ExtraTreeClassifier the best model, I did performance evaluation FLAML's ExtraTreeClassifier using three datasets, imbalanced, and after balancing using Synthetic Minority Over-sampling Technique (SMOTE) and SMOTE + Edited Nearest Neighbors (SMOTEENN). The results is as shown in Table 1.

	Accuracy	**Precision3**	**Recal**	**F1-score**	**ROC-AUC**
Imbalance	0.999	0.817	0.684	0.744	0.966
Smote	0.999	0.712	0.857	0.778	0.947
Smoteenn	0.998	0.535	0.867	0.661	0.974

Table 1: Performance evaluation FLAML's ExtraTreeClassifier using three datasets, imbalanced, and after balancing using Synthetic Minority Over-sampling Technique (SMOTE) and SMOTE + Edited Nearest Neighbors (SMOTEENN)

FLAML's ExtraTreeClassifier exhibits exceptional performance across datasets that are imbalanced, as well as those balanced using SMOTE and SMOTEENN techniques. On the imbalanced dataset, the model achieves a high accuracy of 0.999, reflecting its ability to accurately classify majority class instances. However, it shows limitations in recall (0.684), indicating it may miss some true anomalies due to the skewed class distribution.

When the dataset is balanced using SMOTE (Synthetic Minority Over-sampling Technique), FLAML's ExtraTreeClassifier maintains a high accuracy of 0.999, similar to the imbalanced scenario. The significant improvement comes in recall, which increases to 0.857. This enhancement suggests that SMOTE effectively synthesizes new instances in the minority class, thereby improving the model's ability to detect true anomalies. The F1-score also increases to 0.778, striking a better balance between precision and recall compared to the imbalanced dataset. Despite a slight decrease in precision (0.712), the model maintains strong discriminatory ability with an ROC-AUC score of 0.947.

Balancing the dataset using SMOTEENN (SMOTE + Edited Nearest Neighbors) further enhances FLAML's ExtraTreeClassifier's performance. While the accuracy slightly decreases to 0.998 compared to the imbalanced and SMOTE-balanced scenarios, SMOTEENN significantly improves recall to 0.867. This approach combines oversampling of the minority class with cleaning of noisy samples, resulting in heightened sensitivity to true anomalies. The model achieves an impressive ROC-AUC

score of 0.974, indicating excellent discriminatory power. However, there is a trade-off with precision (0.535), which decreases due to increased false positives.

To extend the analysis, I have generated confusion matrix for each case as show below.

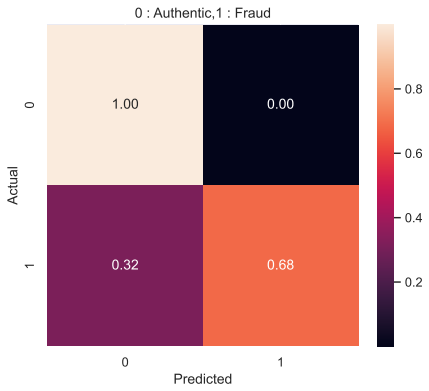


Figure 3: Imbalance Data

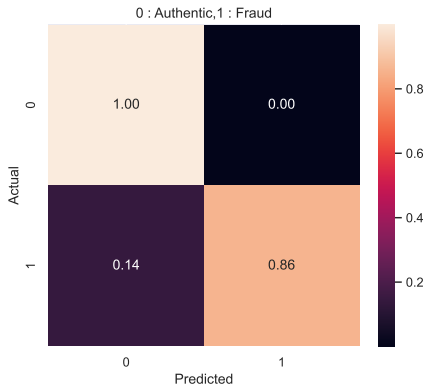


Figure 4: Balance by SMOTE

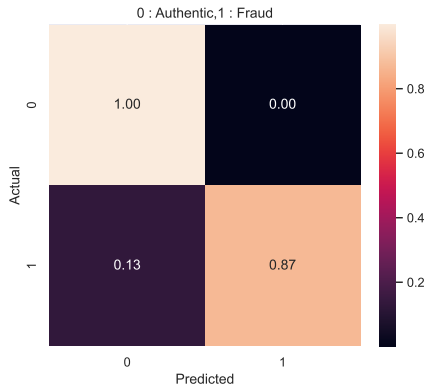


Figure 5: Balance by SMOTEENN

In the context of an imbalanced dataset, the FLAML's ExtraTreeClassifier model exhibits exceptional accuracy (0.999), largely driven by its ability to correctly identify majority class instances (authentic transactions). However, the challenge lies in detecting minority class instances (fraudulent transactions), where the model shows limitations in recall (0.684) and precision (0.817). This imbalance is reflected in the confusion matrix, which highlights a high True Negative rate (1.00) but a comparatively lower True Positive rate (0.32), underscoring the difficulty in effectively capturing fraudulent activities amidst skewed data distributions.

When the dataset is balanced using SMOTE, FLAML's ExtraTreeClassifier shows marked improvement in detecting fraudulent transactions. The model achieves a higher True Positive rate (0.14) compared to the imbalanced scenario, resulting in an enhanced recall (0.857) and a balanced F1-score (0.778). The confusion matrix confirms a maintained high True Negative rate (1.00) for authentic transactions, indicating consistent performance in correctly identifying majority class instances while benefiting from SMOTE's synthetic oversampling to address class imbalance. This approach demonstrates SMOTE's effectiveness in enhancing the model's sensitivity to detecting anomalies, thereby improving overall performance metrics such as ROC-AUC (0.947), which signifies robust discrimination between classes post-oversampling.

Conclusion

In evaluating various anomaly detection models using credit card transaction data, FLAML's ExtraTreeClassifier has emerged as a standout performer across multiple critical metrics. With an impressive accuracy of 0.999, FLAML's model showcases its ability to accurately classify the majority of transactions, maintaining a high precision of 0.817 and a commendable recall of 0.684. These metrics underscore its robustness in identifying both authentic transactions and anomalies with minimal misclassifications. The model's strong ROC-AUC score of 0.966 further validates its discriminatory power in distinguishing between normal and fraudulent transactions, solidifying its suitability for sensitive financial environments.

Comparatively, other models like Local Outlier Factor (LOF) and One-Class SVM (OCSVM) exhibit limitations, particularly in their recall performance in imbalanced datasets. LOF, despite achieving a high accuracy of 0.989, struggles with a low recall of 0.0, indicating challenges in accurately identifying fraudulent transactions. Similarly, while Histogram-based Outlier Score (HBOS) demonstrates competitive performance with an F1-score of 0.241 and a robust ROC-AUC of 0.967, it requires careful consideration in terms of precision-recall balance. Models such as Isolation Forest (IForest) and Minimum Covariance Determinant (MCD) show strong recall rates above 0.8, highlighting their effectiveness in detecting true anomalies despite potential trade-offs in precision. Ultimately, selecting the most appropriate anomaly detection model should align with specific operational needs and data characteristics to optimize performance and mitigate risks effectively in financial settings.

P33 Global Data Lab Weekly Report

Kipkemoi Vincent

July 20, 2024

Objective

In this work, I have used an imbalanced dataset of credit card frauds (with the target labels being authentic and fraudulent) to assess:

- Local and global explainability of different anomaly detectors in predicting anomalies in credit card transactions.
- Where anomaly detectors agree on predictions, do they give rise to similar LIME and SHAP value explanations? How about when they disagree?

Data Description and Feature Selection

The dataset contains information on the transactions made using credit cards by European cardholders, in two particular days of September 2013 . It presents a total of 284807 transactions, of which 492 were fraudulent. Clearly, the dataset is highly imbalanced, the positive class (fraudulent transactions) accounting for only 0.173% of all transactions. The columns in the dataset are as follows:

Time: The time (in seconds) elapsed between the transaction and the very first transaction

V1 to V28: Obtained from principle component analysis (PCA) transformation on original features that are not available due to confidentiality

Amount: The amount of the transaction

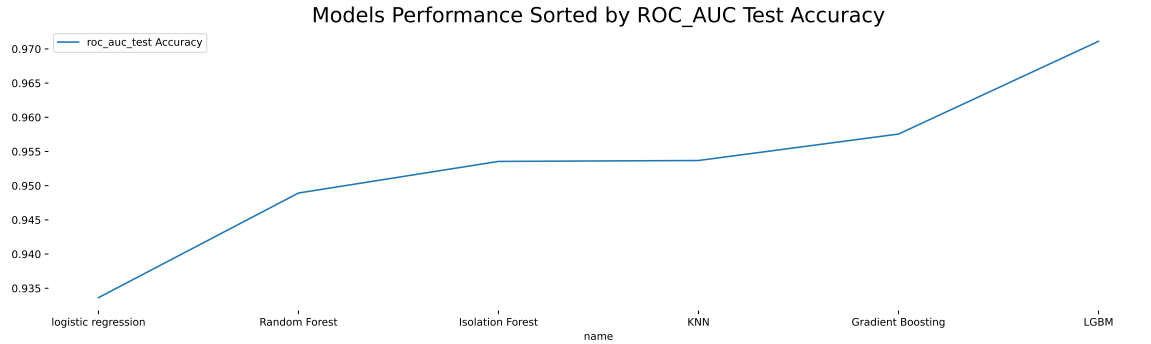
Class: The status of the transaction with respect to authenticity. The class of an authentic (resp. fraudulent) transaction is taken to be 0.

The overall dimension of this dataset after performing some feature engineering is 34. To increase the computational efficiency in calculating LIME and Shap values, I leveraged on three feature selection approaches; Recursive Feature Elimination (RFE), SelectFromModel (SFM) and BorutaPy, to reduce the dimensionality from 34 to 10 by selecting the top 10 features.

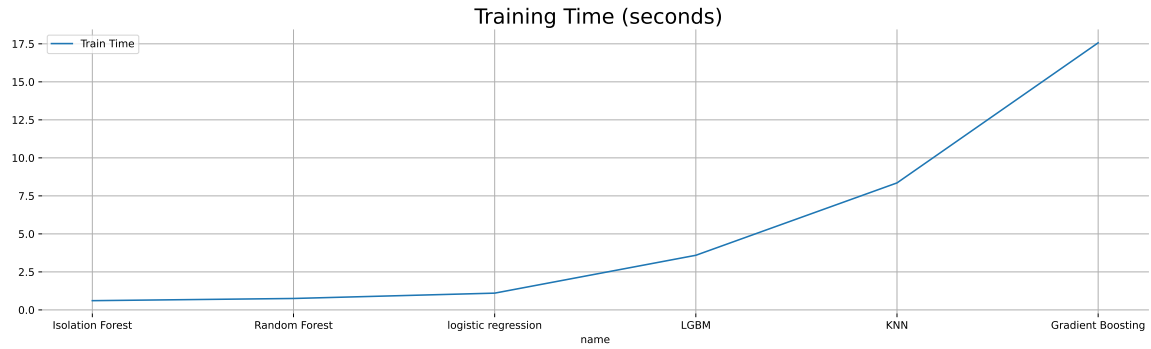
Explaining Models

To obtain explanation for each feature contributions, 6 anomaly detectors were trained (Logistic regression, Random forest, Gradient Boosting , LGBM Classifier , and two PyOD models; KNN and Isolation Forest). We assessed these explanations in two perspectives: global perspective (i.e., how does each feature impact outcomes on the average for the entire datasheet) or in local perspective (i.e., how does each feature impact predictions for a specific instance). Some models have inbuilt properties that provide these sorts of explanations. These are typically referred to as white box models and examples include logistic regression (model coefficients) and Tree-based models (feature importance). Due to their complexity, other models - such as Neural Networks etc. - do not have straightforward methods for explaining their predictions. For these models, (also known as black box models), approaches such as Local Interpretable Model-agnostic Explanation **(LIME)** and SHapley Additive exPlanations **(SHAP)** can be applied.

Figure 1 shows the performance of the trained models in terms of ROC-AUC scores on the testing set and their training time. The results shows that the LGBM Classifier is performing best both in terms of ROC-AUC score and the training time. Even though the KNN and Gradient Boosting models have good ROC-AUC scores, their training times are relative high. The Random Forest and Isolation Forest Models have small training time and hence viable in obtaining explanations in methods that are computational expensive in terms of time such as SHAP calculations.



(a)



(b)

Figure 1: (a) ROC-AUC scores of the the models ion the test data together with (b) their training time.

Global Explanation

0.1 Global Explanation : Logistic Regression Coefficients

Logistic regression is considered one of the white box models as we can infer the contribution of its features by using their respective coefficients. This gives us some idea of how an increase/change in each feature might result in prediction capability of the model. We can also get a general understanding of how important a feature is for the entire dataset by looking at its magnitude of its coefficient relative to the magnitude of the other coefficients.

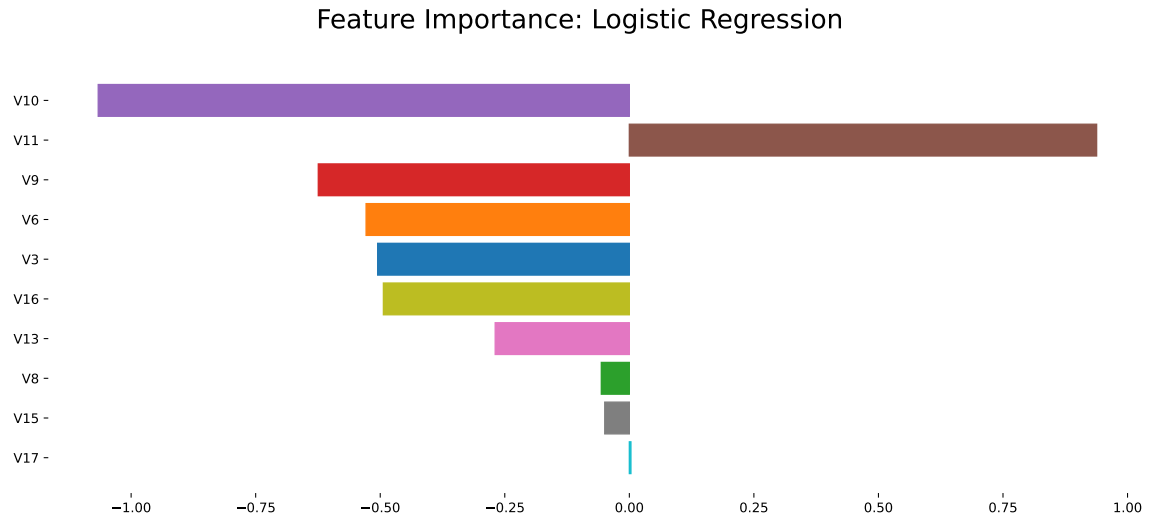


Figure 2: Feature Importance obtained using Logistic regression model.

Figure 2 shows the feature importance obtained using logistic regression anomaly detector for the credit card transaction dataset. From the results, feature, 'V10' contribute more while feature', 'V17' contribute the lowest, on the prediction capability of the logistic regression model. Except feature, 'V11', all the other features contribute negatively on the prediction

of the model.

0.2 Global Explanation : Feature Importance Score vs Permutation Feature Importance Scores

Here we compare the global feature importance obtained using the feature importance function which is build within tree based models and permutation feature importance function build within the sklearn. The feature importance function compute the mean decrease in impurity for each feature - i.e., how impactful the feature is in reducing the uncertainty (classifiers) or variance (regressors) of the decision tree prediction. This value is also known as the gini importance score.

Permutation feature importance work by evaluating how much the model's performance decreases when the values of a specific feature are randomly shuffled, effectively breaking the relationship between that feature and the target variable. The decrease in model performance after shuffling indicates the importance of the feature: the larger the decrease, the more important the feature is considered.

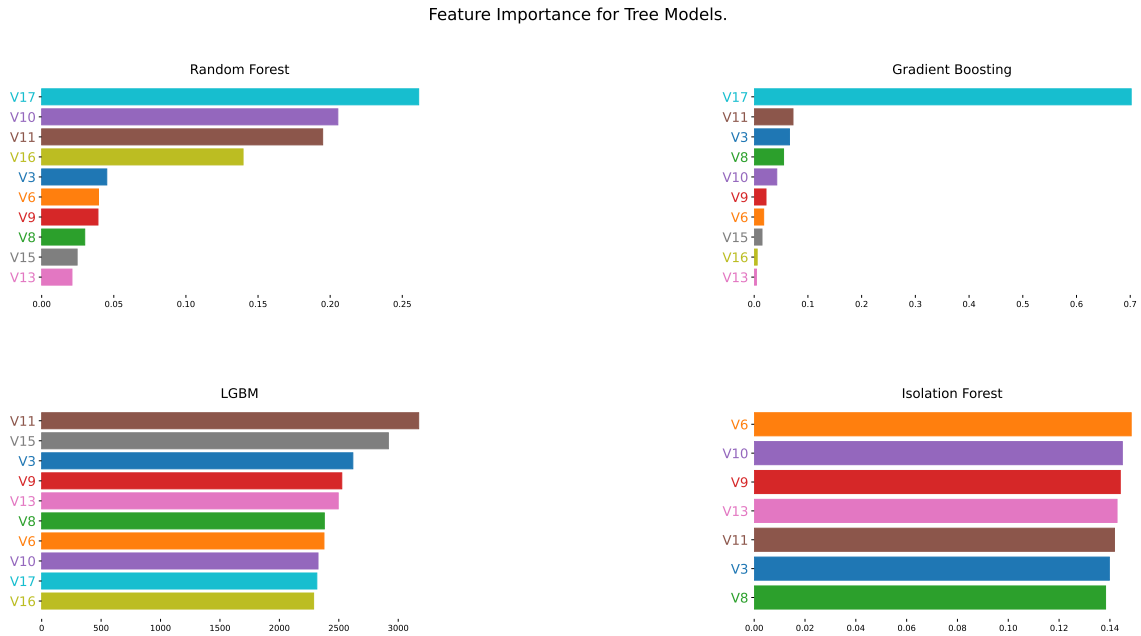


Figure 3: Feature Importance of the tree models obtained using inbuilt feature importance function

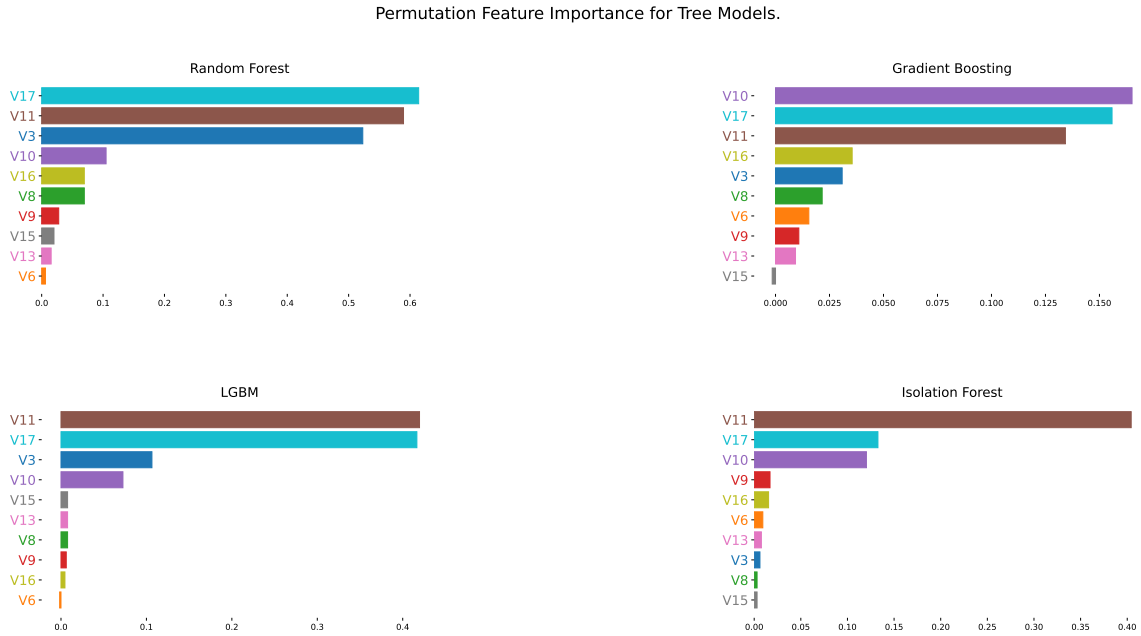


Figure 4: Feature Importance of the tree models obtained using the permutation feature importance build within sklearn

Permutation feature importance excels by providing a model-agnostic approach, unlike built-in feature importance functions tied to specific algorithms. It effectively handles multicollinearity and non-linear relationships, aiding robust feature selection and engineering tasks. Moreover, its computational efficiency makes it ideal for large datasets and complex models, contrasting with potentially slower built-in methods such as those relying on SHAP values.

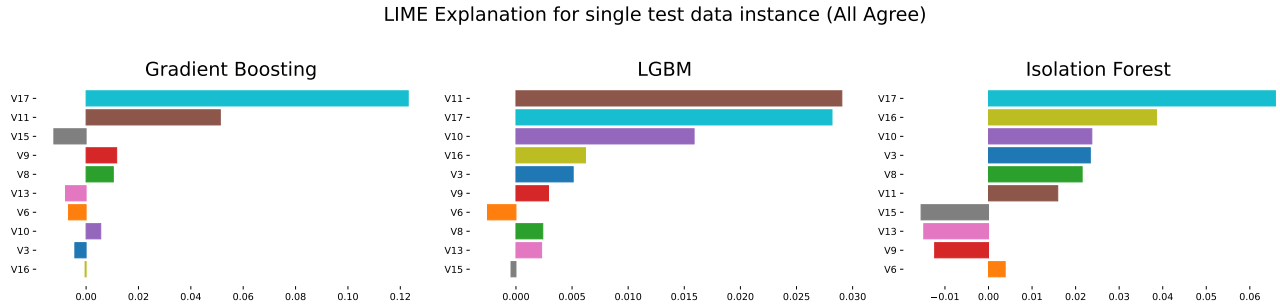
Comparing Fig. 3 and Fig. 4, features, 'V10', 'V11' and 'V17' are consistently in top 4 contributors in Fig. 4 compared to that in Fig. 3. This highlight that the permutation feature importance is robust in calculating feature contribution when compared to the in-build feature importance function.

Local Explanation

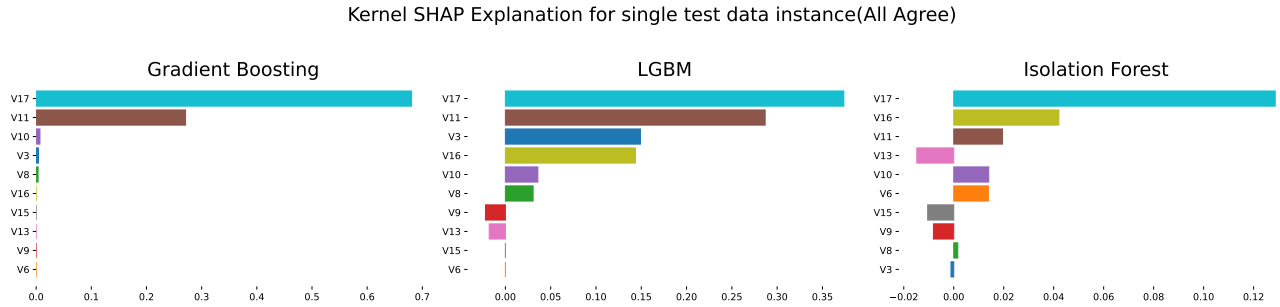
To assess the local explainability (explainability at a particular instance in a dataset), I have used three detectors; Gradient Boosting, Isolation Forest and LGBM Classifier and, considered LIME and SHAP explanations at three instances:

- Where all the three models agree in correctly predicting fraudulent transactions
- Where only Gradient Bosting and Isolation Forest agree in correctly predicting fraudulent transactions
- Where only Isolation Forest and LGBM Classifier agree in correctly predicting fraudulent transactions

0.3 Case where all the three models agree in correctly predicting fraudulent transactions



(a) LIME

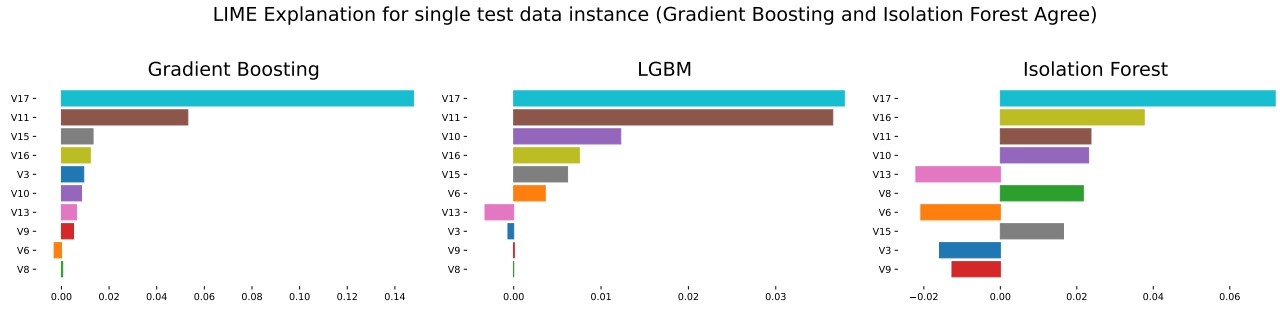


(b) SHAP

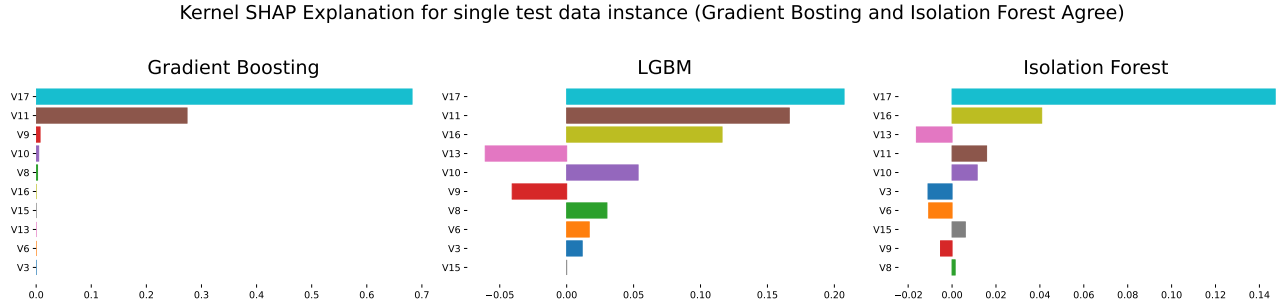
Figure 5: Explainability using (a) LIME and (b) SHAP values calculated at an instant where all the three models agree in correctly predicting fraudulent transactions.

Figure 5 shows the feature contribution in terms of LIME and SHAP values for each of the three models at a particular instance in the test set where prediction of all the three models agree in correctly predicting fraudulent transaction. For this particular instance, features, 'V10','V11', 'V16' and 'V17' are generally the key contributors with 'V17' being the key contributor. On whether a feature contribute positively or negatively, the LIME and SHAP explanations seems to agree well for Isolation Forest, which is not the case for the other two models (Gradient Boosting and LGBM Classifier). It is also worth noting that even though prediction agreed for this particular instance, the LIME and SHAP contributions features across the models are relatively different. This highlight the fact that the prediction architecture across different models might be different.

0.4 Case where only Gradient Bosting and Isolation Forest agree in correctly predicting fraudulent transactions



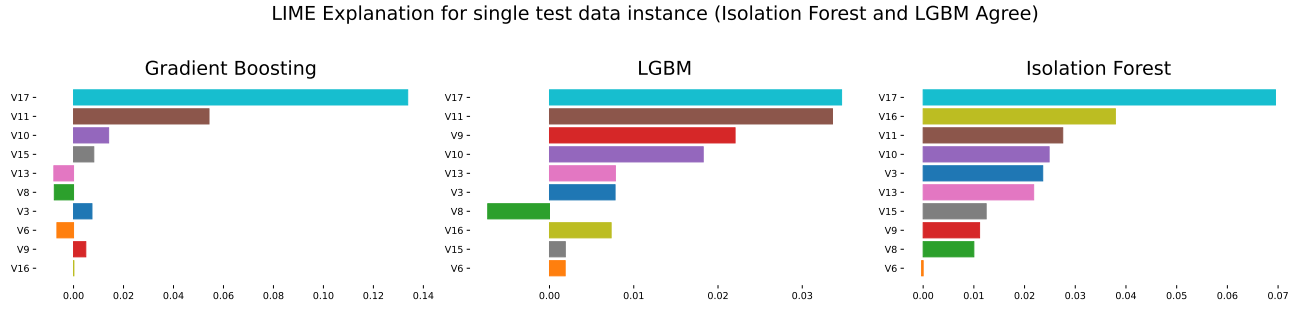
(a) LIME



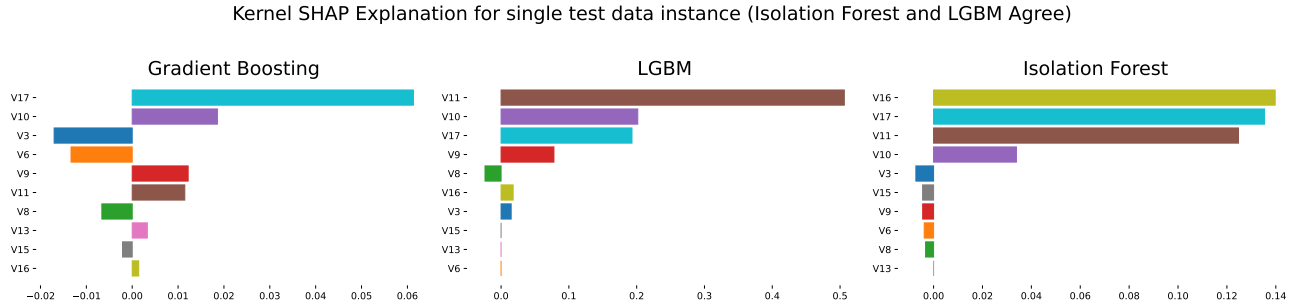
(b) SHAP

Figure 6: Explainability using (a) LIME and (b) SHAP values calculated at an instant where only Gradient Bosting and Isolation Forest agree in correctly predicting fraudulent transactions

0.5 Case where only Isolation Forest and LGBM Classifier agree in correctly predicting fraudulent transactions



(a) LIME



(b) SHAP

Figure 7: Explainability using (a) LIME and (b) SHAP values calculated at an instant where only Isolation Forest and LGBM Classifier agree in correctly predicting fraudulent transactions

Figure 6 and Figure 7 shows the feature contribution in terms of LIME and SHAP values for each of the three models at a particular instance in the test set only two detectors agree in correctly predicting fraudulent transactions. The results seems to indicate feature, 'V17' was the top contributing feature, regardless of whether the model was correctly predicting the fraudulent transaction in the dataset.

Conclusion

In conclusion, this study demonstrated the effectiveness of different anomaly detection models in predicting credit card fraud using an imbalanced dataset. It highlighted the importance of both global and local explanations in understanding model predictions, particularly in scenarios where model transparency is crucial. The use of feature selection techniques and interpretability tools such as LIME and SHAP provided insights into feature contributions, aiding in trustworthiness and interpretability of model outcomes. Future research could explore ensemble methods or hybrid approaches to further enhance both predictive accuracy and explainability in fraud detection systems.

References

M. A. Ahmed, A. M. Atiya, N. El Gayar, and H. El-Shishiny, "Interpretable Models for Healthcare Using Feature Importance Techniques," Procedia Computer Science, vol. 65, pp. 897-904, 2015.

C. Molnar, "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable," 2019.

J. Brownlee, "Permutation Feature Importance in Python," Machine Learning Mastery, 2020.

S. Rashidi, "Explaining Machine Learning Models with Feature Importance," Towards Data Science, 2021.

T. V. Pham, "Interpretable Machine Learning with Python: Explaining the LIME," Towards Data Science, 2019.

. Sengupta, "Interpretable Machine Learning: Introduction to SHAP Values," Towards Data Science, 2020.

P33 Global Data Lab Weekly Report

Kipkemoi Vincent

July 26, 2024

Objective

In this work, I have used two datasets: credit data and census data to evaluate and compare the computational efficiency (execution time) of 3 tree explainer algorithms:

- **TreeShap** algorithm build within the SHAP package
- **FastTreeShap v1** and **FastTreeShap v2** algorithms build within the FastTreeShap package as defined in Lundberg, S. M., & Lee, S. I. (2017). The two algorithms are modifications of TreeShap to fully allow parallel computing.

From Lundberg, S. M., & Lee, S. I. (2017), the time complexity of a tree detector in calculating shap values is a function of a number of variables including the (1) number of samples used, (2) number of estimators tree estimators and (3) maximum depth of each each tree.

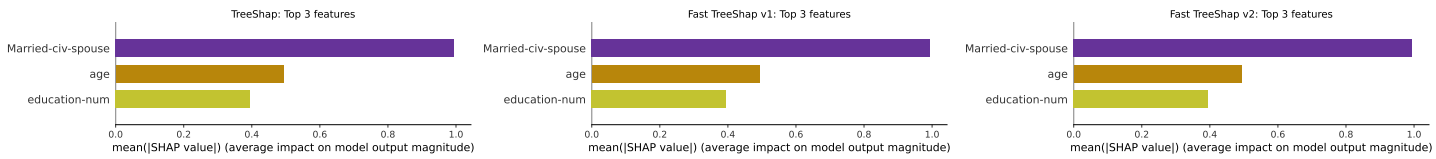
Using Random Forest (RF) and Isolation Forest (IF) detectors and varying these variables, I have examined their execution times when calculating SHAP values using TreeShap, FastTreeShap v1 and FastTreeShap v2 algorithms.

Table 1 shows the description of the datasets. The number of instances in the credit transaction datasets was higher than 200000. To make the SHAP calculations tractable in a reasonable time, I created a sub-sample of 100000 instances for this dataset.

Table 1: Dataset description

Dataset	# Instances	# Attributes (original)	# Attributes (feature engineering)
Credit	100,000 (sub-sample)	30	34
Census	48,882	14	64

Prior to comparing the three SHAP algorithms in terms of execution times, we compared the shap values calculated by FastTreeShap v1 and FastTreeShap v2, against those calculated by baseline algorithm (TreeShap), and found that the maximum difference is insignificant (lower than 10^{-7}). Figure 1 shows the top 3 feature ranking based on shap values for RF model implemented on Census data. The results shows that the shap calculations based on TreeShap, FastTreeShap v1 and FastTreeShap v2 algorithms are similar and provides similar explainability.

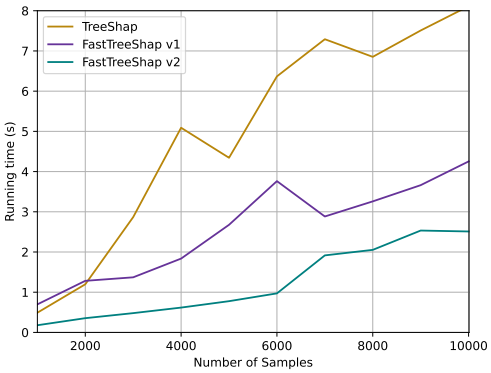


(a) TreeShap (b) FastTreeShap v1 (c) FastTreeShap v2

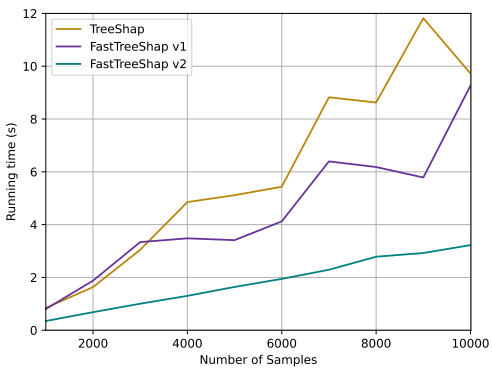
Figure 1: RF SHAP calculations on census data obtained using the three algorithms

Varying the number of samples

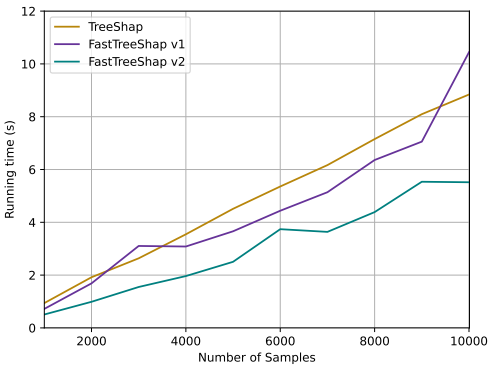
Figure 2 shows the execution times for each of the SHAP algorithms when the number of samples are varied from 1000 to 10000. Note that to effect these calculations, the number of estimators is set to 100 for both models. The maximum depth parameter is set to 8 for the RF model while for IF and based on its documentation, it is already fixed at ceil($log_2(n)$), where n is the number of samples.



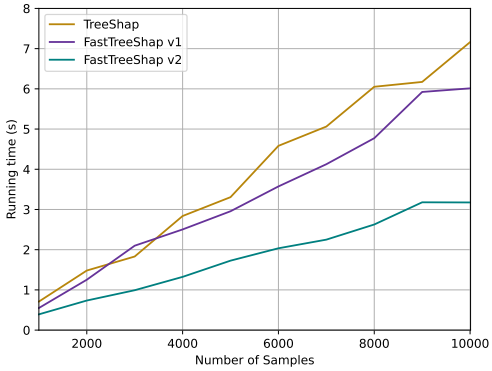
(a) Credit data (RF)



(b) Credit data (IF)



(c) Census data (RF)

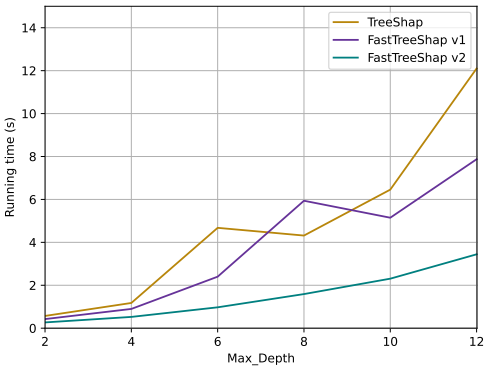


(d) Census data (IF)

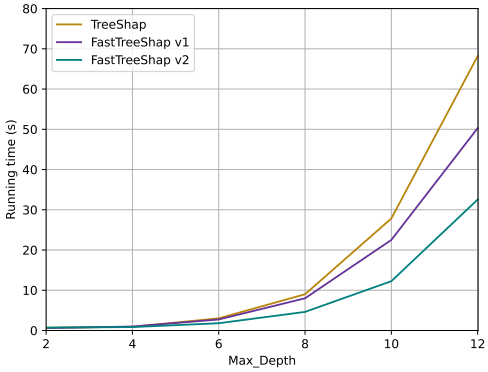
Figure 2: The running times for RF and IF Shap calculations using the three algorithms; TreeShap, FastTreeShap v1 and FastTreeShap v2 when the number of samples is varied from 1000 to 10000

The results shows that SHAP calculations via FastTreeShap v1 and FastTreeShap v2 are generally faster compared that of TreeShap. The speed up in SHAP calculations is more pronounced for FastTreeShap v2 compared to FastTreeShap v1, especially at higher sample size.

Varying the maximum depth



(a) Credit data



(b) Census data

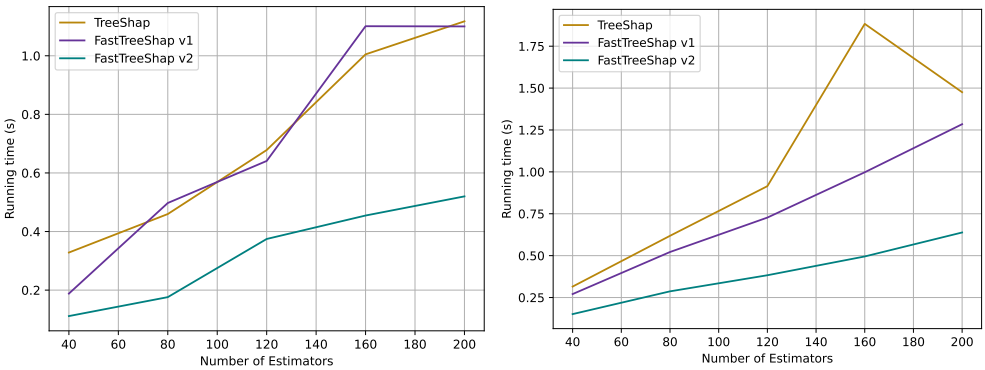
Figure 3: The running times for RF Shap calculations using the three algorithms; TreeShap, FastTreeShap v1 and FastTreeShap v2 when the maximum depth is varied from 2 to 12.

As initially mentioned, the maximum depth of IF is fixed at ceil($log_2(n)$). Here I only examined the effect RF's maximum depth in SHAP calculation execution for the two datasets. To do this, I have fixed the number of samples at 10000 and set the RF number of estimators at 100.

The result in Fig. 3 shows that at lower values of maximum depth (<4), the difference in the execution times for the three algorithms seems to be minimal. As the values of maximum depth increases the SHAP calculations execution times generally grows exponentially. The speed up for FastTreeShap v2 is more pronounced than that of FastTreeShap v1 especially at higher values of maximum depth.

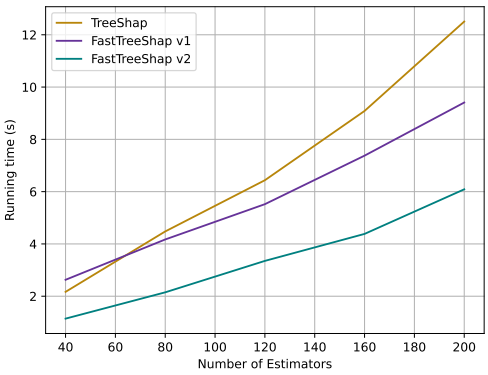
Varying the number of estimators

To assess the effect of model's number of estimators oh SHAP calculation execution time, I fixed the number of sample at 10000 for the two models and maximum depth at 8 for the RF model. Figure 4 shows the execution time the three algorithms when the number of estimators each model is varied from 40 to 200. It is worth noting that the speed up for FastTreeShap v2 is significantly higher compared to that of FastTreeShap v1, especially when the number of estimator is set at high values. At smaller values of number of estimators (< 60), SHAP calculations using the TreeShap seems to be generally faster than calculations using FastTreeShap v1. As the number of estimators increases beyond 60, however, the execution time for FastTreeShap v1 compared to that of TreeShap.

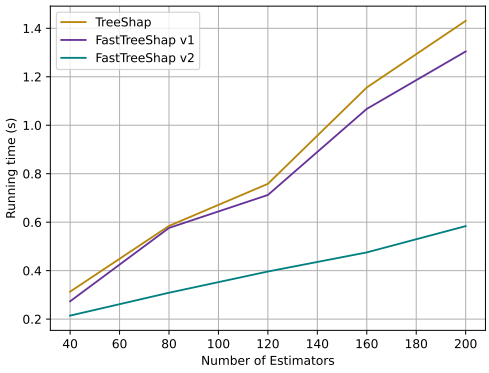


(a) Credit data (RF)

(b) Credit data (IF)



(c) Census data (RF)



(d) Census data (IF)

Figure 4: The running times for RF and IF Shap calculations using the three algorithms; TreeShap, FastTreeShap v1 and FastTreeShap v2 when the number of estimators in each model is varied from 40 to200

Summary

Table 2 illustrates the average speedup achieved by FastTreeShap v1 and FastTreeShap v2 in computing SHAP values for Credit and Census data. Depending on the dataset and the variable of concern, the average speed up for FastTreeShap v1 range from 1.08 to 1.84, while that of FastTreeShap v2 range from 1.67 to 4.58.

Table 2: Average speed up

Dataset	Variable	Speed up (RF)		Speed up (IF)	
		FastTreeShap v1	FastTreeShap v2	FastTreeShap v1	FastTreeShap v2
Credit	No. of samples	1.84	4.58	1.29	3.15
	Maximum depth	1.35	3.03	-	-
	No. of estimators	1.13	2.35	1.29	3.15
Census	No. of samples	1.12	1.69	1.16	2.07
	Maximum depth	1.15	1.67	-	-
	No. of estimators	1.12	2.01	1.08	2.03

Conclusion

The comparison of SHAP value computation algorithms revealed that FastTreeShap v2 significantly accelerates execution times compared to TreeShap and FastTreeShap v1. Specifically, FastTreeShap v2 achieved up to a 4.58x speedup over TreeShap for the Credit dataset and 2.07x for the Census dataset. This improvement is most notable with increased sample sizes, deeper tree depths, and more estimators. The results underscore FastTreeShap v2's superior efficiency, making it highly suitable for large-scale and complex model analyses. Overall, FastTreeShap v2 offers substantial computational advantages, enhancing practical applicability in machine learning tasks.

References

Lundberg, S. M., & Lee, S. I. (2017). Fast TreeSHAP: Accelerating SHAP Value Computation for Trees. Proceedings of the 34th International Conference on Machine Learning (ICML)

C. Molnar, "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable," 2019.

Sengupta, "Interpretable Machine Learning: Introduction to SHAP Values," Towards Data Science, 2020.