Cairo University-Faculty of Engineering
Computer Department
Machine Learning

Large-scale Energy Anomaly Detection (LEAD)

Machine Learning Project

Team 6

Full Name	ID	SEC	BN
اسامة صالح فرج السيد	9210195	1	12
عبدالرحمن محمد عبدالفتاح محمود	9210587	1	23
عبدالرحمن محمد حفني	9210584	1	22
عمرو صلاح الدين فؤاد	9210774	1	30

Submitted to:

Eng. Mohamed Shawky

2025

Problem Definition

Create a machine learning model to identify anomalies in hourly smart electricity meter readings, distinguishing between normal consumption patterns and unusual energy usage.

Problem Motivation

Detecting energy consumption anomalies is crucial for energy providers to prevent fraud, identify faulty meters, and promote energy efficiency. Early detection can lead to significant cost savings and more reliable energy distribution.

Evaluation Metrics

- **Accuracy**
- **Precision & Recall**
- **F1-score**
- **Area Under the ROC** (AUC-ROC)

Dataset Link and information

[Link]

Dataset has 1.75M rows and 57 features, missing values, and imbalanced classes with ‘0’ class spanning 97% of the dataset.

Exploratory Data Analysis (EDA)

While exploring our dataset we did the following:

1) Removed features that have only one value

Features ***gte_meter***, ***year*** were removed.

2) Exploring features' relations

- Features like ***hour***, ***day***, ***week***, ***year***, ***hour_x***, **...** are just an expanded form of ***timestamp*** feature, so we removed ***timestamp***.
- Also, features like ***building_weekday***, ***building_...*** are just combination of ***building_id*** and features mentioned above, so we also removed them.

3) Handling Null Values

Only ***meter_reading*** feature had null values, and all rows that have ***meter_reading*** as null are not anomalous (***anomaly*** = 0).

We dropped those rows as our dataset is large enough anyway.

4) Handling non-numeric features

Only ***primary_use*** feature is non-numeric and has 12 unique categorical values.

We used One-Hot Encoding to convert it to 11 Boolean features.

5) Exploring features' correlations

Highly correlated features were removed.

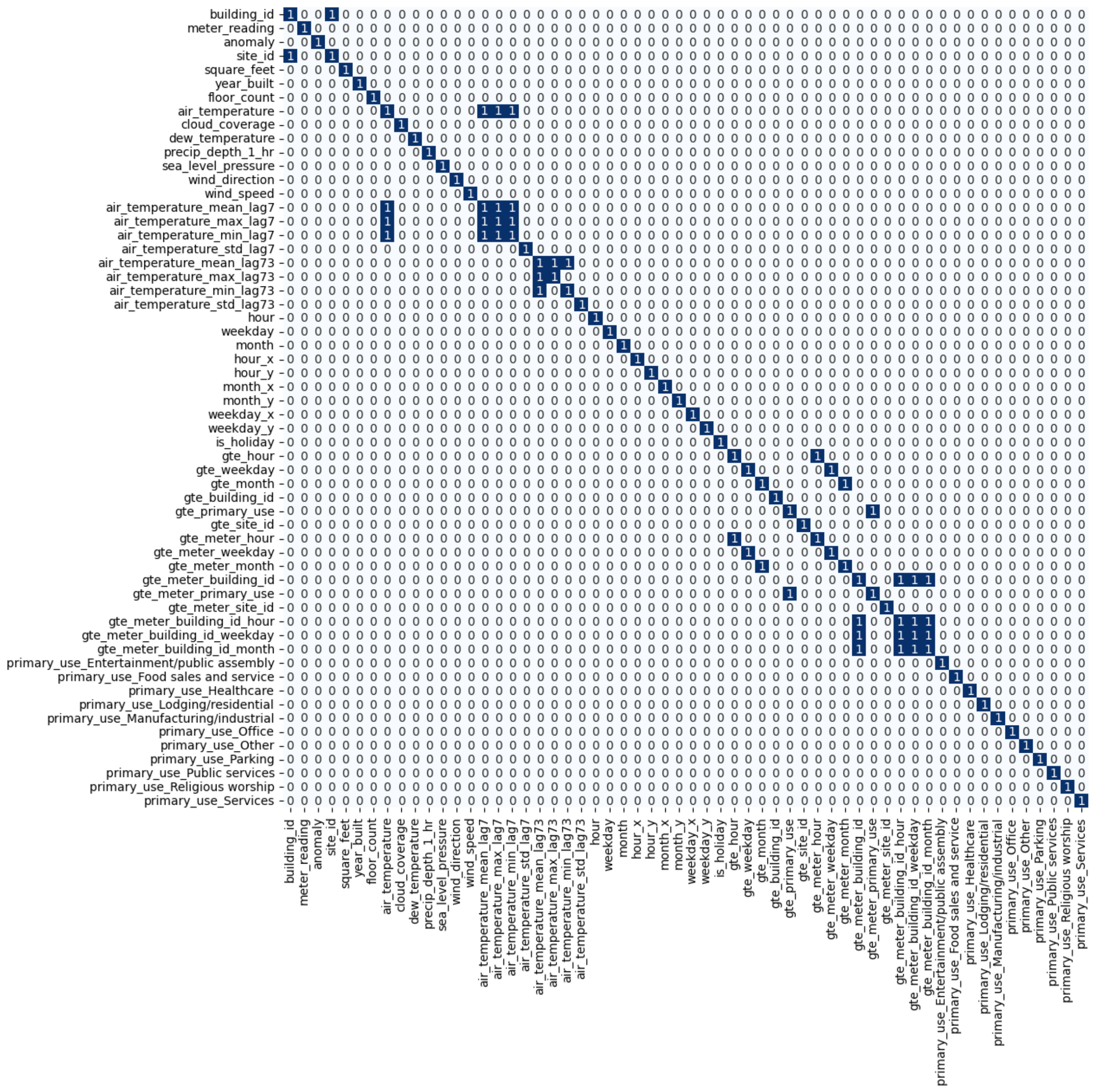
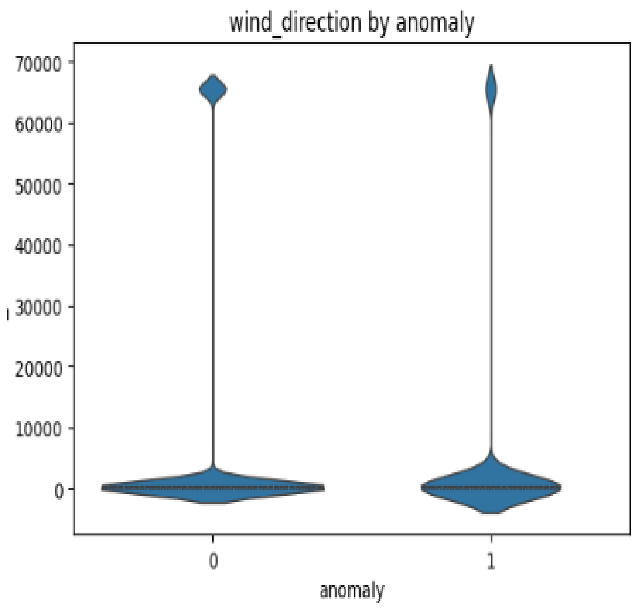


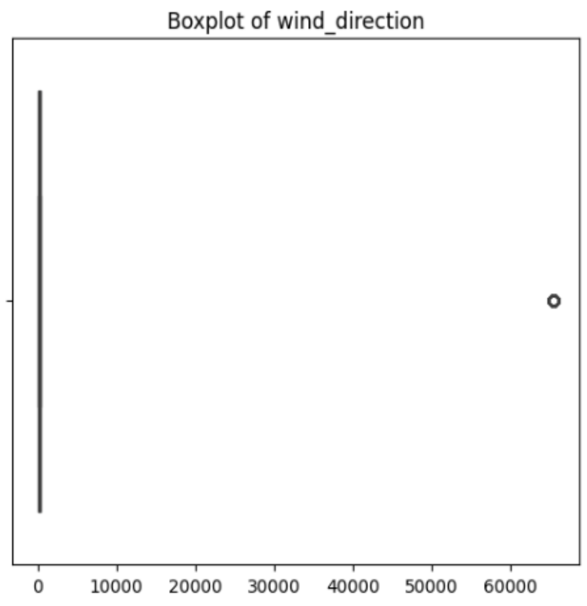
Figure 1: Correlation between features with threshold > 0.95

6) Handling Outliers

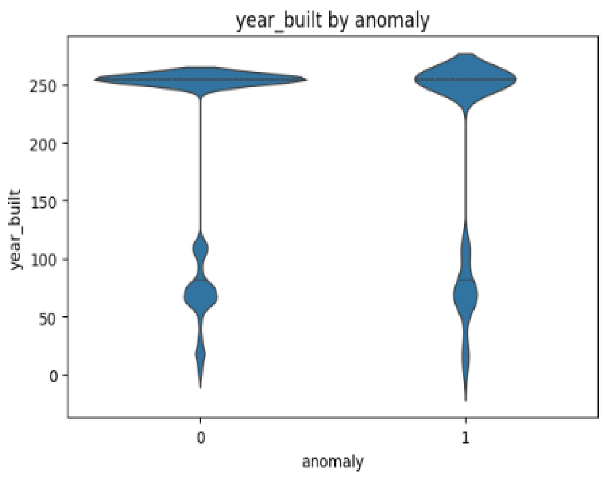
Features ***meter_reading***, ***year_built***, ***cloud_coverage***, ***percip_depth_1_hr***, ***wind_direction*** had outliers that were handled:

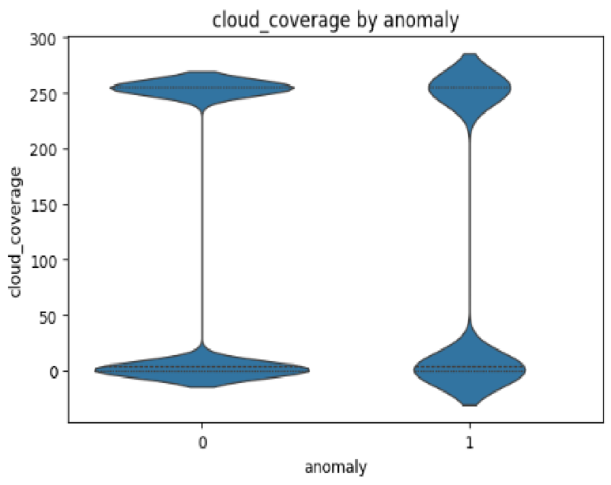
- ***wind_direction*** outliers were handled by using log transformation.



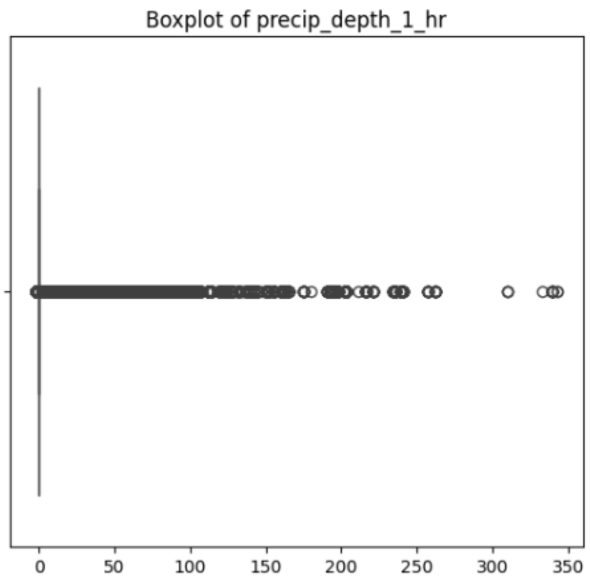


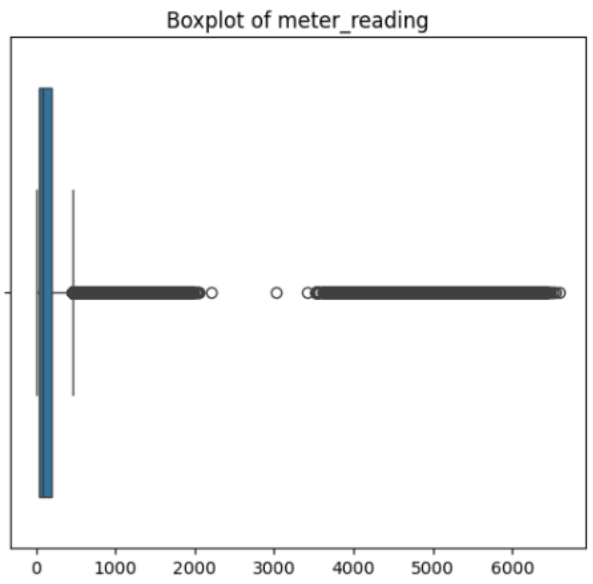
- ***year_built*** and ***cloud_coverage*** outliers were just a replacement value for missing values that spanned a large amount of the dataset, we dropped them as they aren't related to energy consumption anomalies anyway.





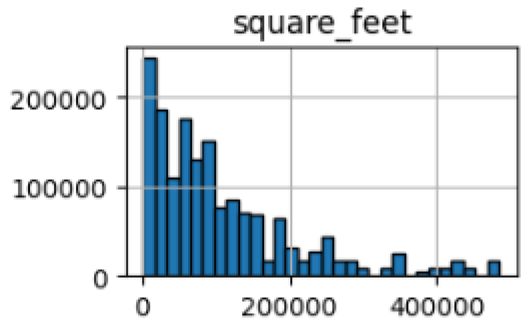
- ***meter_reading*** and ***percip_depth_1_hr*** outliers were handled by using min-max scaling.

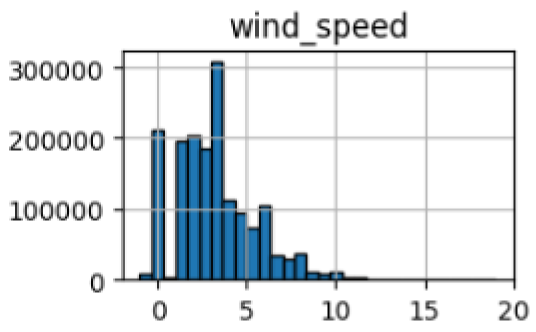




7) Other Observations

- Some features like ***building_id, site_id, hour, weekday, month, hour_x, hour_y, month_x, month_y, weekday_x, weekday_y, is_holiday, gte_hour, gte_weekday, gte_month, floor_count*** aren't really related to if there is an anomaly in energy consumption or not. They may be useful, so we will try both using and not using them.
- The features' distributions of ***square_feet*** and ***wind_speed*** were right-skewed, so we applied log transformation to them.





Experiments and Results

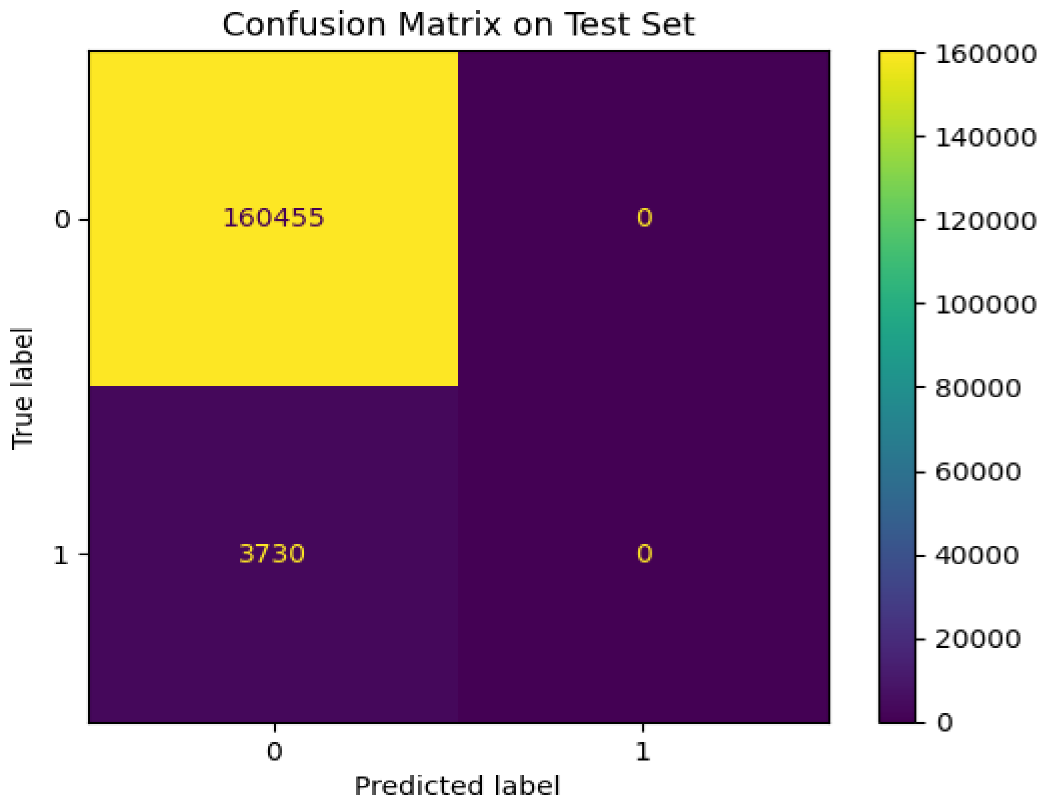
In all experiments, we will try to surpass **ZeroR** base model that gives 97% accuracy.

Dataset was split into train and test sets using stratification on ***anomaly***, so each set has the same distribution of ***anomaly*** feature.

1) Linear Model (Logistic Regression)

We tried applying Logistic Regression with stochastic gradient descent learning as a linear binary classifier.

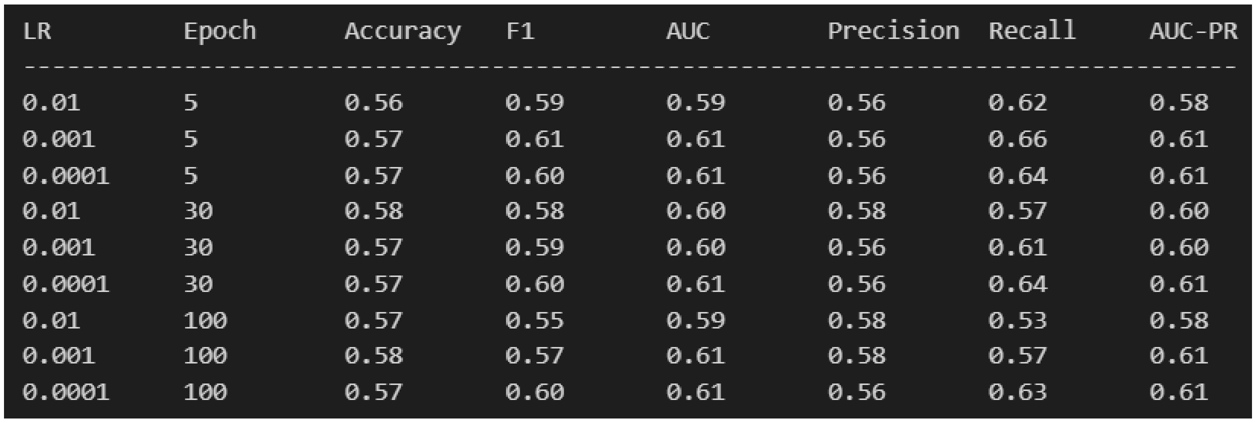
The results were bad as the dataset is heavily imbalanced and the model couldn't find a linear separator to separate data



So, we tried to handle the imbalance of the dataset using two approaches:

- **Down Sampling**

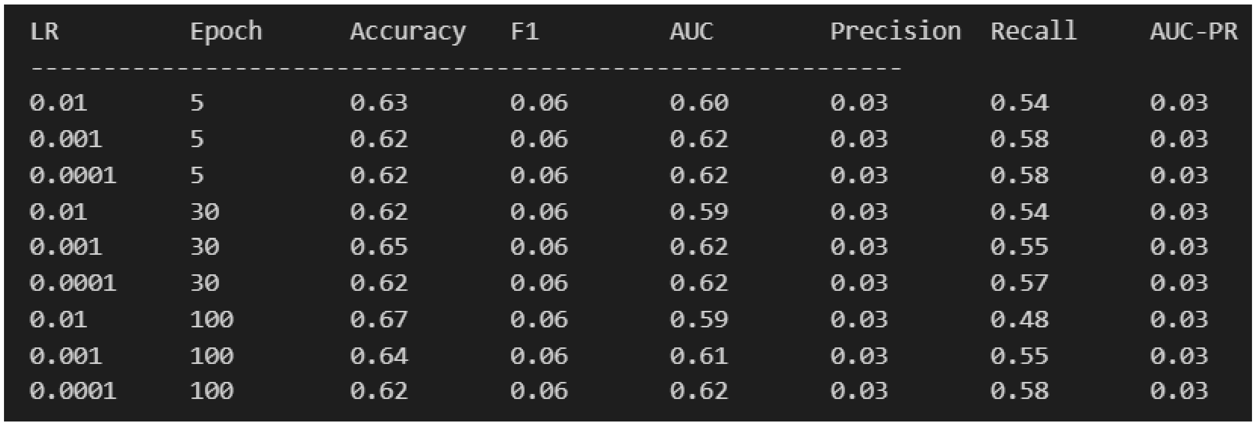
We used a balanced subset of the dataset with 50% coverage for both classes, trained the model with different numbers of epochs and different learning rates.



```
LR          Epoch       Accuracy    F1          AUC         Precision   Recall      AUC-PR
------------------------------------------------------------------------------------------
0.01        5           0.56        0.59        0.59        0.56        0.62        0.58
0.001       5           0.57        0.61        0.61        0.56        0.66        0.61
0.0001      5           0.57        0.60        0.61        0.56        0.64        0.61
0.01        30          0.58        0.58        0.60        0.58        0.57        0.60
0.001       30          0.57        0.59        0.60        0.56        0.61        0.60
0.0001      30          0.57        0.60        0.61        0.56        0.64        0.61
0.01        100         0.57        0.55        0.59        0.58        0.53        0.58
0.001       100         0.58        0.57        0.61        0.58        0.57        0.61
0.0001      100         0.57        0.60        0.61        0.56        0.63        0.61
```

- **Over Sampling using SMOTE (Synthetic Minority Over-sampling Technique)**

By generating synthetic data from the minor class (***anomaly*** = 1) till the dataset is balanced, we also trained the model with different numbers of epochs and different learning rates.



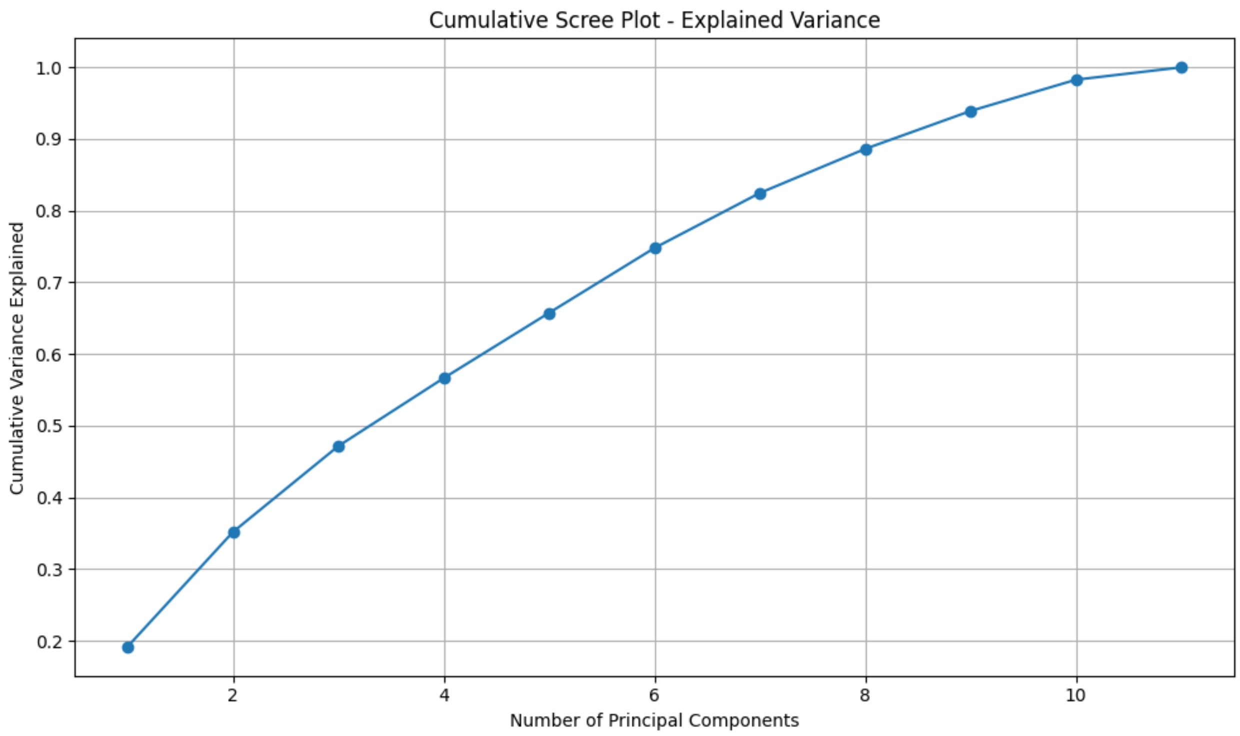
```
LR          Epoch       Accuracy    F1          AUC         Precision   Recall      AUC-PR
----------------------------------------------------------------
0.01        5           0.63        0.06        0.60        0.03        0.54        0.03
0.001       5           0.62        0.06        0.62        0.03        0.58        0.03
0.0001      5           0.62        0.06        0.62        0.03        0.58        0.03
0.01        30          0.62        0.06        0.59        0.03        0.54        0.03
0.001       30          0.65        0.06        0.62        0.03        0.55        0.03
0.0001      30          0.62        0.06        0.62        0.03        0.57        0.03
0.01        100         0.67        0.06        0.59        0.03        0.48        0.03
0.001       100         0.64        0.06        0.61        0.03        0.55        0.03
0.0001      100         0.62        0.06        0.62        0.03        0.58        0.03
```

We can see that the model still underfits with the dataset balanced, meaning our data can't be linearly separated.

The model achieved similar results when using features removed in EDA.

2) Non-Linear Model (SVM)

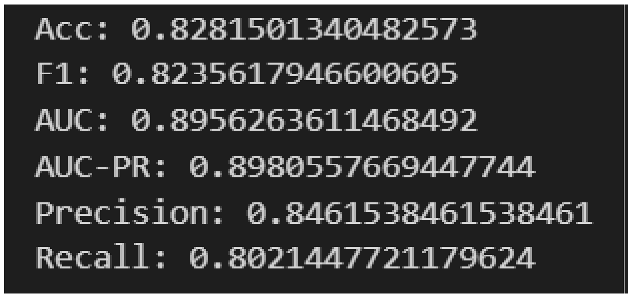
As Non-linear SVM takes a lot of time in training, we will first apply PCA to see if we can reduce the dimensionality of our feature space. PCA was applied on the continuous features only.



We can see that almost all features contribute to the variance and dropping a lot of them will affect model's performance badly.

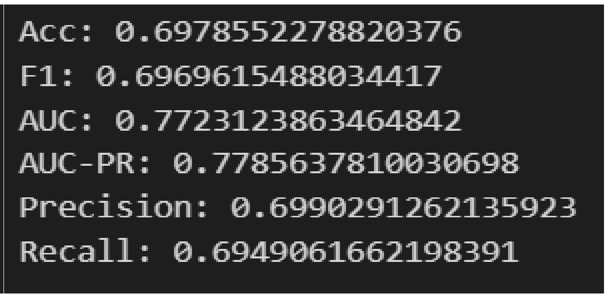
So, we tried two other approaches using SVM with a gaussian kernel (RBF):

- **Down Sampling** (same as linear model)



```
Acc: 0.8281501340482573
F1: 0.8235617946600605
AUC: 0.8956263611468492
AUC-PR: 0.8980557669447744
Precision: 0.8461538461538461
Recall: 0.8021447721179624
```

Figure 2: Non-Linear SVM



```
Acc: 0.6978552278820376
F1: 0.6969615488034417
AUC: 0.7723123863464842
AUC-PR: 0.7785637810030698
Precision: 0.6990291262135923
Recall: 0.6949061662198391
```

Figure 3: Non-Linear SVM with features removed

- **RBF Kernel approximation**

By transforming our feature space into a linear one by approximating the gaussian kernel using these steps:

1) Sample D random vectors *ω*i from the Fourier transform of the gaussian kernel

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

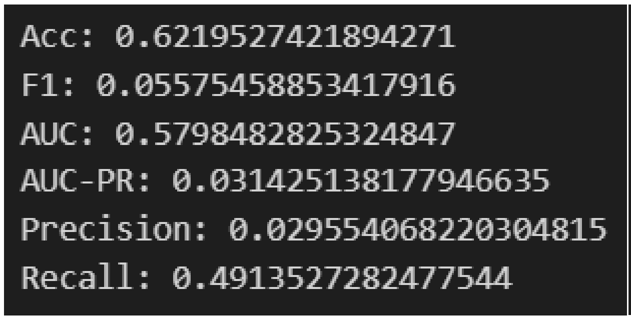
2) Sample random phases *b*i from Uniform [0, 2pi]
3) Using this feature mapping

$$z(x) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos(w_1^T x + b_1) \\ \cos(w_2^T x + b_2) \\ \vdots \\ \cos(w_D^T x + b_D) \end{bmatrix}$$

4) Use Dot Products in the New Space

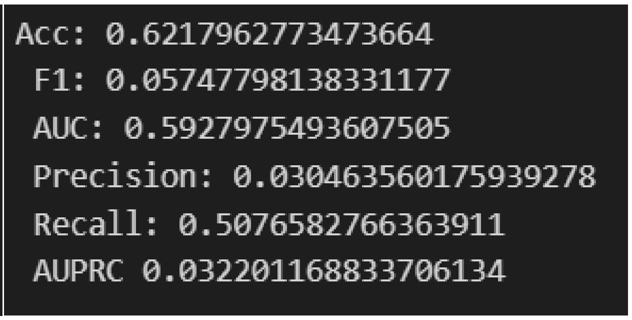
$$K(x, x') \approx z(x)^T z(x') = \sum_{i=1}^{D} \frac{2}{D} \cos(w_i^T x + b_i) \cos(w_i^T x' + b_i)$$

Then we applied a linear SVM on this linear space and got the following results:



```
Acc: 0.6219527421894271
F1: 0.05575458853417916
AUC: 0.5798482825324847
AUC-PR: 0.031425138177946635
Precision: 0.029554068220304815
Recall: 0.4913527282477544
```

Figure 4: On Test set



```
Acc: 0.6217962773473664
 F1: 0.05747798138331177
 AUC: 0.5927975493607505
 Precision: 0.030463560175939278
 Recall: 0.5076582766363911
 AUPRC 0.032201168833706134
```

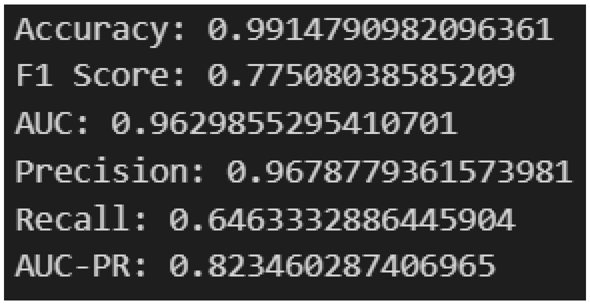
Figure 5: On Train set

We can see that SVM performs better than logistic regression, but it still underfits and linear approximations made it perform worse.

3) Ensemble Model (XGBoost)

We used XGBoost instead of AdaBoost to reduce training time as our dataset is large and XGBoost supports parallelization.

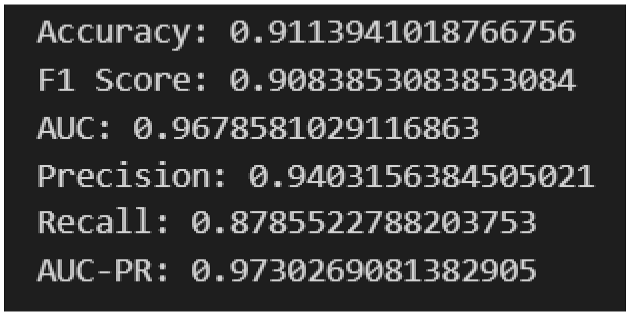
XGBoost achieved good results on the whole dataset:



```
Accuracy: 0.9914790982096361
F1 Score: 0.77508038585209
AUC: 0.9629855295410701
Precision: 0.9678779361573981
Recall: 0.6463332886445904
AUC-PR: 0.823460287406965
```

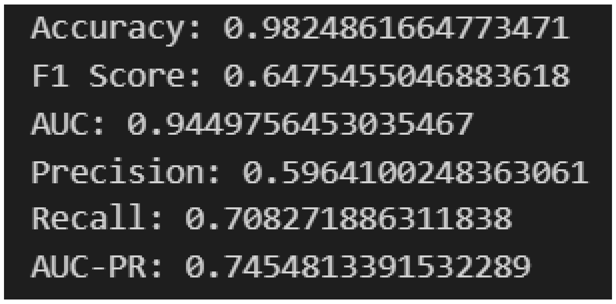
We tried to improve its performance by:

1) Balancing dataset
   - **Down Sampling** (same as linear model)



```
Accuracy: 0.9113941018766756
F1 Score: 0.9083853083853084
AUC: 0.9678581029116863
Precision: 0.9403156384505021
Recall: 0.8785522788203753
AUC-PR: 0.9730269081382905
```

   - **Over Sampling using SMOTE** (same as linear model)



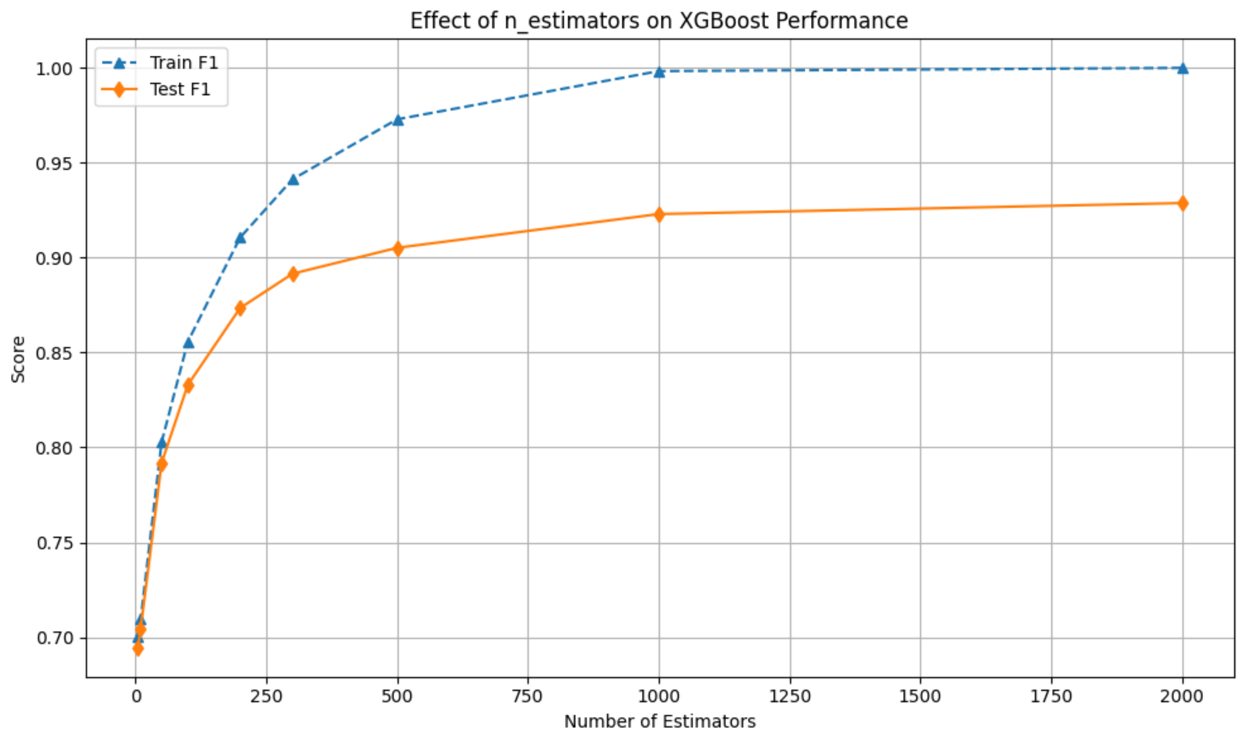
```
Accuracy: 0.9824861664773471
F1 Score: 0.6475455046883618
AUC: 0.9449756453035467
Precision: 0.5964100248363061
Recall: 0.708271886311838
AUC-PR: 0.7454813391532289
```

We can see that balancing the dataset didn't help as

- Taking a small subset of data didn't make the model generalize well
- Generating synthetic data made the model overfit and generalize poorly to the imbalanced test set
- Boosting models are somewhat robust to imbalanced datasets anyway

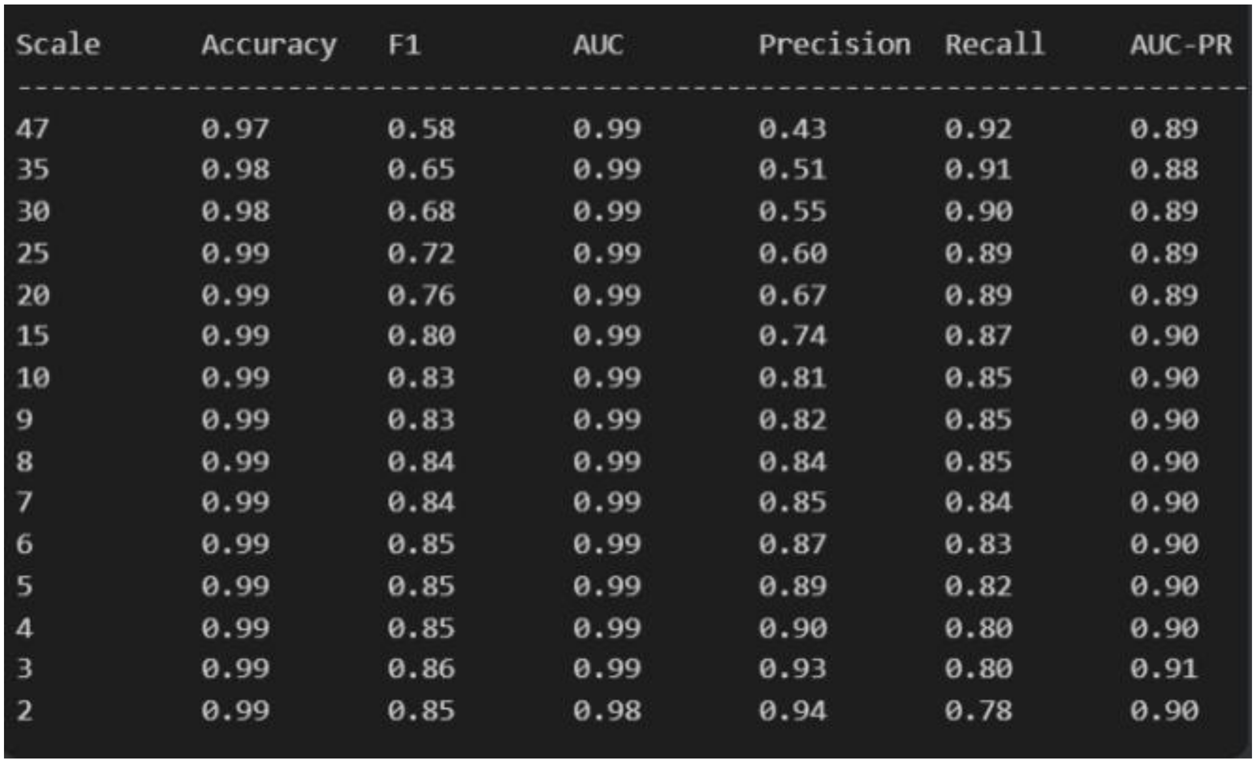
2) Hyperparameter Tuning

  - **Number of Estimators**



We found that the best number of estimators is 1000.

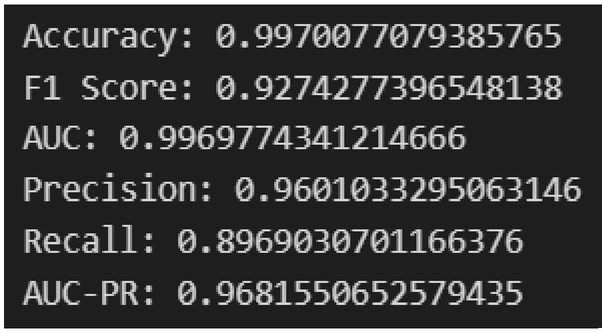
- **Class weighting**



```
Scale       Accuracy    F1          AUC         Precision   Recall      AUC-PR
--------------------------------------------------------------------------------
47          0.97        0.58        0.99        0.43        0.92        0.89
35          0.98        0.65        0.99        0.51        0.91        0.88
30          0.98        0.68        0.99        0.55        0.90        0.89
25          0.99        0.72        0.99        0.60        0.89        0.89
20          0.99        0.76        0.99        0.67        0.89        0.89
15          0.99        0.80        0.99        0.74        0.87        0.90
10          0.99        0.83        0.99        0.81        0.85        0.90
9           0.99        0.83        0.99        0.82        0.85        0.90
8           0.99        0.84        0.99        0.84        0.85        0.90
7           0.99        0.84        0.99        0.85        0.84        0.90
6           0.99        0.85        0.99        0.87        0.83        0.90
5           0.99        0.85        0.99        0.89        0.82        0.90
4           0.99        0.85        0.99        0.90        0.80        0.90
3           0.99        0.86        0.99        0.93        0.80        0.91
2           0.99        0.85        0.98        0.94        0.78        0.90
```

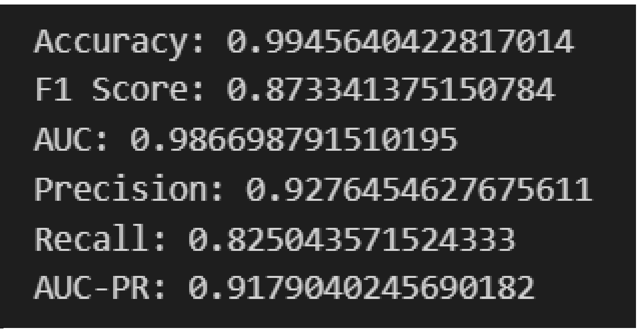
We found that scaling positive class weights by 3 gave the best performance.

After trying the best parameters, we got the following results:



```
Accuracy: 0.9970077079385765
F1 Score: 0.9274277396548138
AUC: 0.9969774341214666
Precision: 0.9601033295063146
Recall: 0.8969030701166376
AUC-PR: 0.9681550652579435
```

Figure 6: Score on test set



```
Accuracy: 0.9945640422817014
F1 Score: 0.873341375150784
AUC: 0.986698791510195
Precision: 0.9276454627675611
Recall: 0.825043571524333
AUC-PR: 0.9179040245690182
```

Figure 7: Score on test set after removing features

Conclusion

Linear model (Logistic Regression) performed badly whether the dataset was balanced or not due to the dataset not being linearly separable.

Non-Linear model (SVM) performed better than linear one but still underfits.

Ensemble model (XGBoost) achieved the best results with accuracy better than our base model ZeroR.

Keeping the features mentioned in EDA gave better results.

Workload Division

Full Name	Task
اسامة صالح فرج السيد	EDA
عبدالرحمن محمد عبدالفتاح محمود	Non-linear Model
عبدالرحمن محمد حفني	Linear Model
عمرو صلاح الدين فؤاد	Ensemble Model