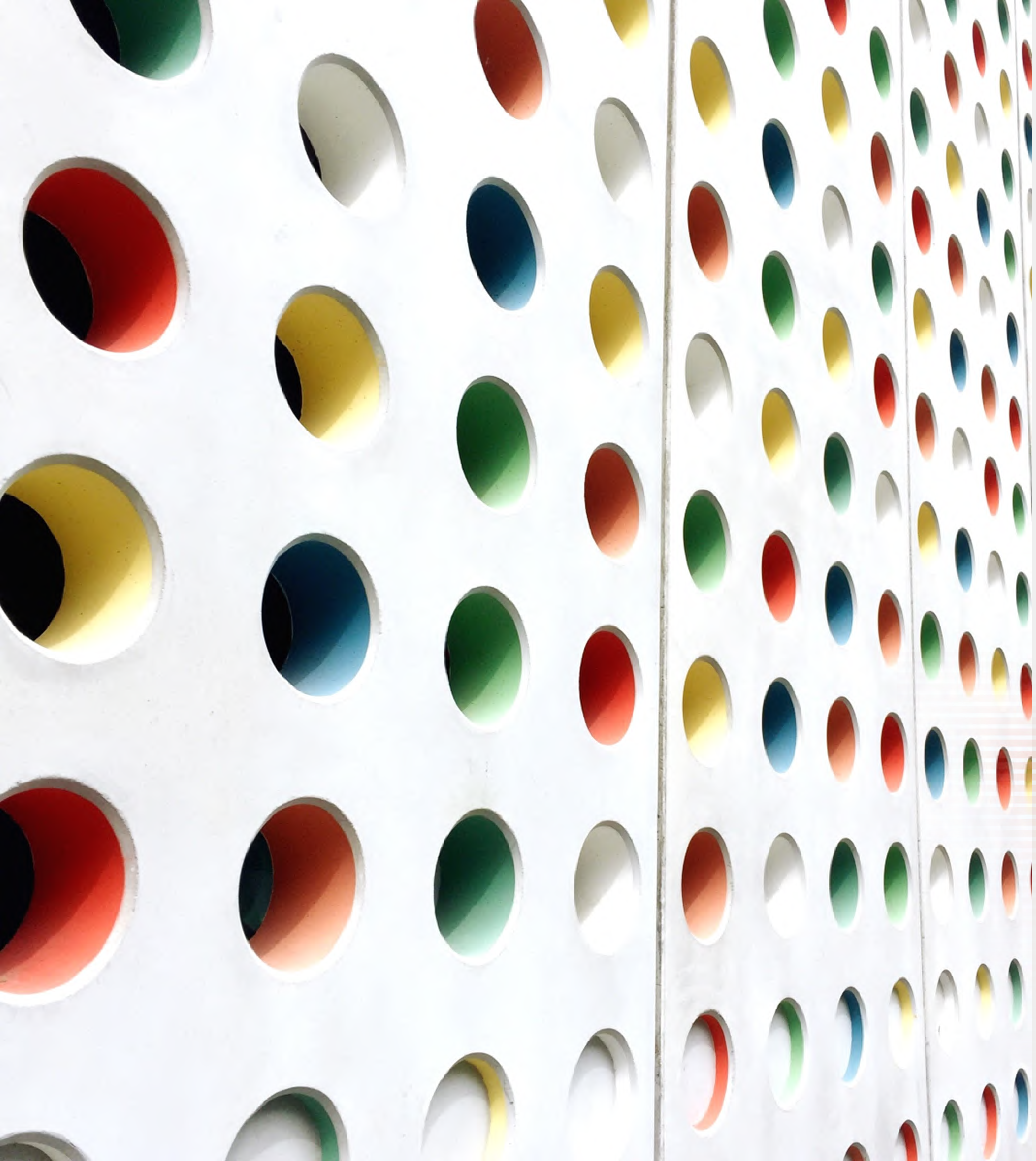


Seeping through the plethora of numbers,

Intelligent Finance Function Predictor

Contents::

- Introduction
- Background and Motivation
- Key methodologies and their WHY's?
- Results and their Analysis
- Insight report and Conclusion

Introduction:

Finance datasets are literally a gigantic matrix of numbers, and what is one of the biggest tech we've invented that revolves only around numbers, machine learning. Then why not use it to deal with its domain?

Problem statement:

Create a mapping function, denoted as $y = f(x)$, using a dataset that includes attributes x1, x2, x3, x4,..., xn. The target variable indicates whether a specific entity has been historically identified as a "0" or "1".

Objective:

The objective given is to construct a predictive model $F\theta(X) \rightarrow Ypred$ that accurately estimates the target variable Y{i} for new, unseen inputs X{i}

Background and Motivation

Financial data sets are complex and voluminous, holding crucial insights for decision-making. that's why traditional methods often fall short, necessitating advanced systems like machine learning.

And that's why,

Background and Motivation

we just had had to:

- **Enhance accuracy and reliability of financial predictions.**
- **Leverage a model capable enough to identify the hidden patterns with the variables unknown to us.**
- **Revolutionize financial data interpretation and utilization, giving space for future advancements.**

So, this project isn't just about numbers; it's about revolutionizing how financial data is interpreted and utilized, driving better business decisions and fostering innovation in financial management.

Key methodologies and WHY?

Stage 1: Column Analysis

We wanted to configure the complexity of this numerical and categorical cacophony. So this is what we did:

- Noticed that column 9 had 93% of missing values, we decided to drop it, as well as the ID column from all the datasets.
- Calculated the range, entropy, number of unique values for each column of the training as well as testing dataset.
- Assigned with a bit of help from domain analysis to really differentiate which column would be categorical and numerical respectively.

Stage 1: Column Analysis

WHY?

- Dropped column 9 so because it could not be imputed, as it would simply introduce errors in the model because of too much uncertainty.
- Range, entropy, number of unique values. For simple overview of the columns, range giving us the minimum and maximum values, entropy giving us the randomness value of the columns, and calculating number of unique values delivered what we needed to differentiate the data.
- Finally, it was important to differentiate because we couldn't possibly scale the potential categorical columns, that would be a bit catastrophic to the model.

Stage 2: Preprocessing

- Pipeline creation: After categorizing numerical and categorical columns, we created a pipeline.
- Mode imputer for categorical columns, Median imputer for numerical ones.
- Made sure the scaling only applied to numerical columns.
- Applying preprocessing to training and testing datasets as well as saving the pipeline.
- Outlier removal through Z-score method. Calculation of the percentage of the data removed as outliers.
- Visualization of numerical and categorical columns separately. Their distributions and their correlation matrix (for the cleaned data).

Stage 2: Preprocessing

WHY?

- Pipeline creation was done for ease of testing of validation data.
- Scaling could not be applied to categorical columns because of creation of misleading relationships.
- Using Z-score method method to be a bit light on the outliers, because the data was really imbalanced.

Stage 3: Feature Engineering

- Defined another pipeline for feature selection + feature interaction.
- Feature selection through SelectKBest method, Feature interaction of only the top 6 numerical columns and adding them to our original 21 feature dataset
- After applying this pipeline to the training and testing datasets we ensured there are no duplicate columns, and the datasets are perfectly aligned eventually saving the pipeline.

Stage 3: Feature Engineering

WHY?

- Feature selection through SelectKBest method, our choice for a perfect method to get things done without the computation time or power hassle.
- Feature interaction of only the top 6 columns or else the number of columns would be too large for machine learning to work on it.

Stage 3: Model Training

- SMOTE: Handle class imbalance, the key to biggest problems in the data was smote for us. Lifesaver, a definite one. Applied to only the training data to balance the minority class.
- Visualization of the balanced data of numerical columns and the correlation matrix.
- PCA: Creation of a pipeline and applying Principal Component Analysis to reduce dimensionality, bringing the data down to 20 features (applied to both training and testing data).
- Visualization of the components retained, and saving the pipeline.

Stage 3: Model Training

- Model selection: Applied in a different file, we got a rough idea of which models we could choose. The options being Logistic Regression, KNN, SVM, Random Forest, Naive Bayes Classifier, Decision Tree and Gradient Boosting as well as XGBoost.
- Creation of a stacking classifier, base models being XGBoost and SGDClassifier and the meta model being something simpler and saving the pipeline of the model.

Stage 3: Model Training

- Model selection to choose the best model we could considering the computation aspect, we didn't go with auto ml, manually chose the best models.
- Creation of a stacking classifier was done to leave room for incremental learning (covering the upgrade of the model aspect), and to really get the best estimates that uncover underlying linear or non-linear patterns.

WHY?

Stage 4: Model Optimization

Hyperparameter Tuning:
We used hyperparameter tuning on both our base models to get the best parameters. GridSearch on SGDClassifier and RandomSearch on XGBoost, considering the time and power.

And another aspect we covered was to only tune the hyperparameters on the subset of the data instead of the entire data considering again, computing time.

WHY?

Stage 5: Model Evaluation

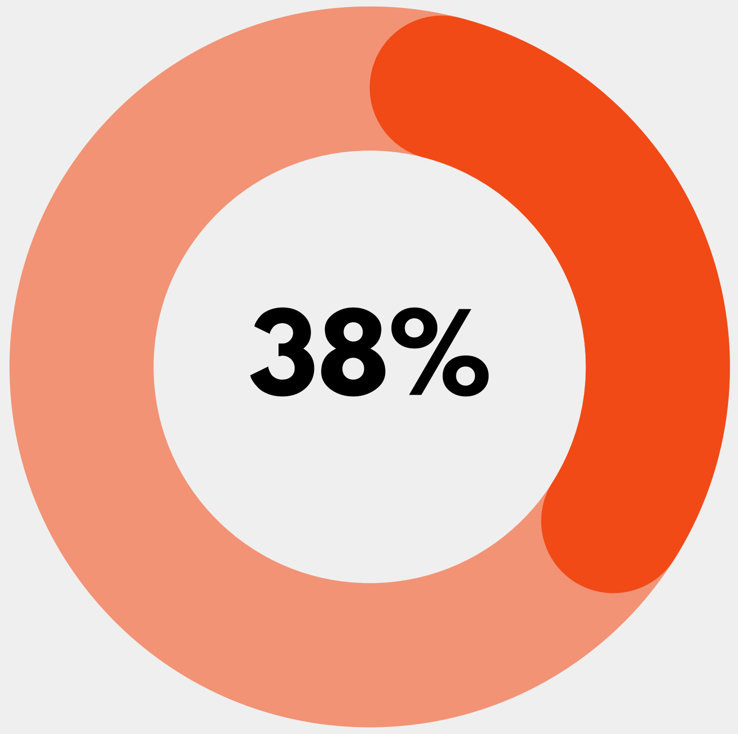
- Used Stratified K-fold Cross Validation method (the why - ensuring that each fold has the same proportion of classes as the entire dataset .
- Defining y_pred as our prediction target value set.
- Confusion matrix, Classification Report (accuracy, precision, recall, F1 score) as well as AUC-ROC curve.

Stage 6: Interpretation of the Results

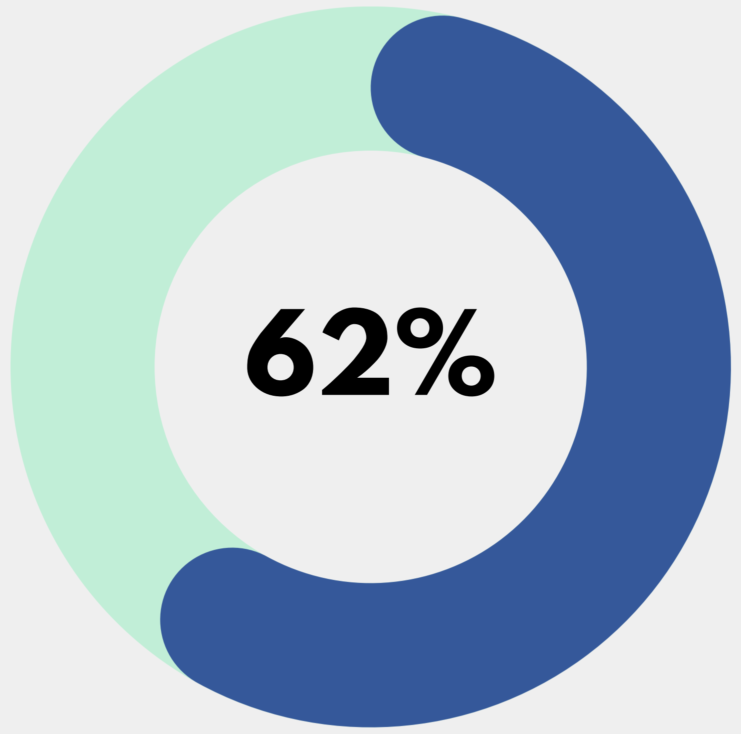
- Trained XGboost's best and SGDClassifier's best models (before we trained the meta model only as the best_estimator).
- Visualized PCA components using shap on both models.
- Inverse pca values to the original columns (with interaction terms).
- Visualization of the variables' effect on our final prediction.

Commented code for testing validation dataset.

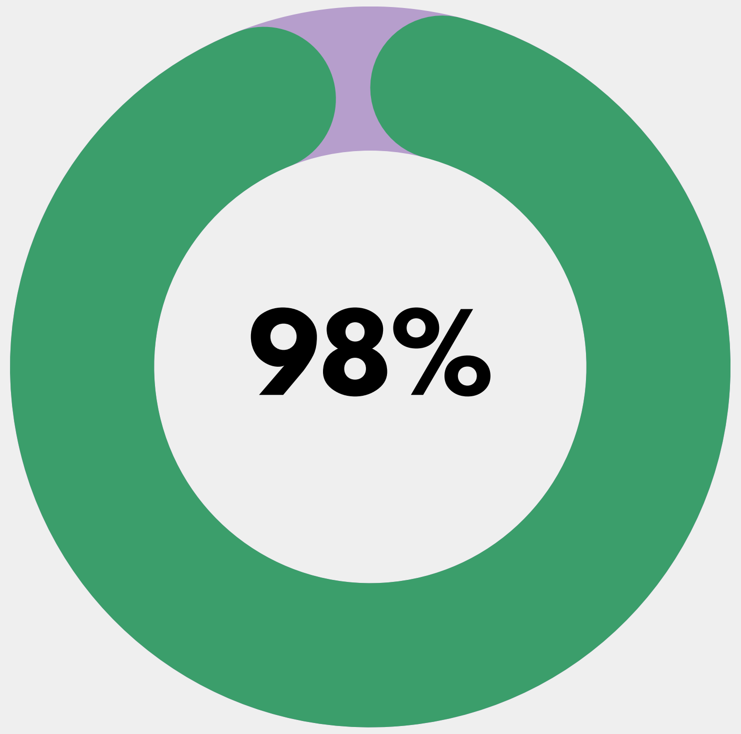
Model Specific Statistics



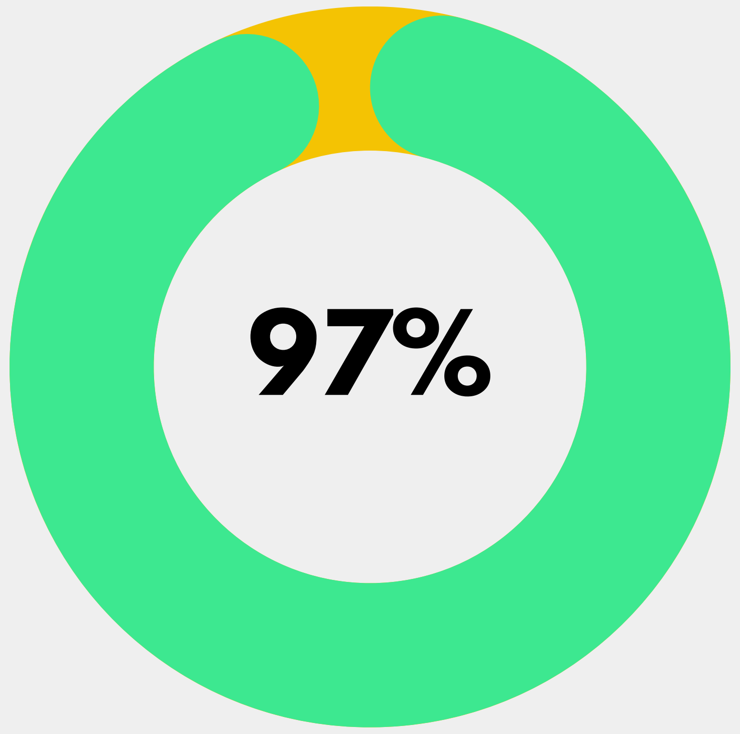
Numerical columns



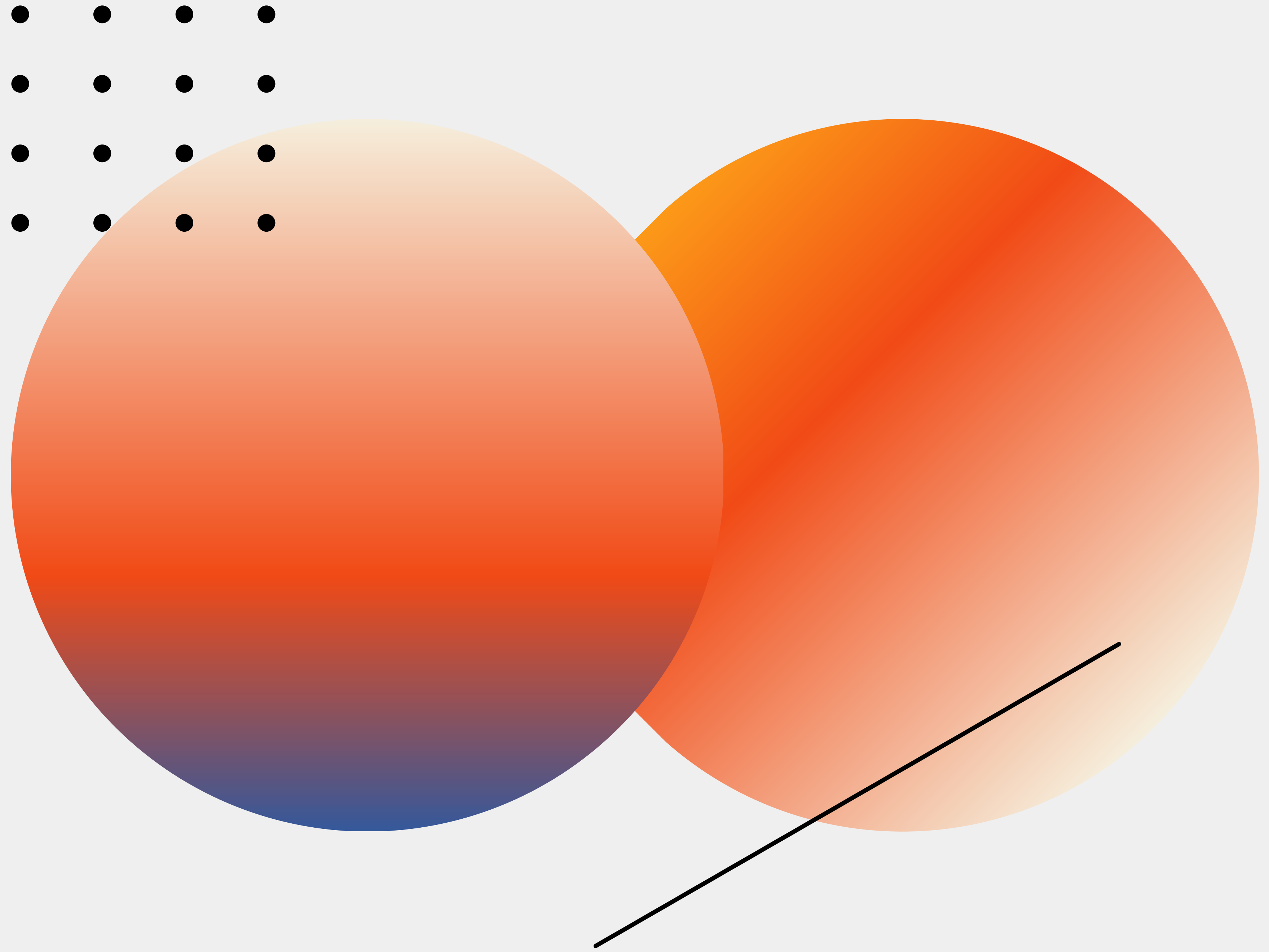
Categorical Columns



PCA info retrieval



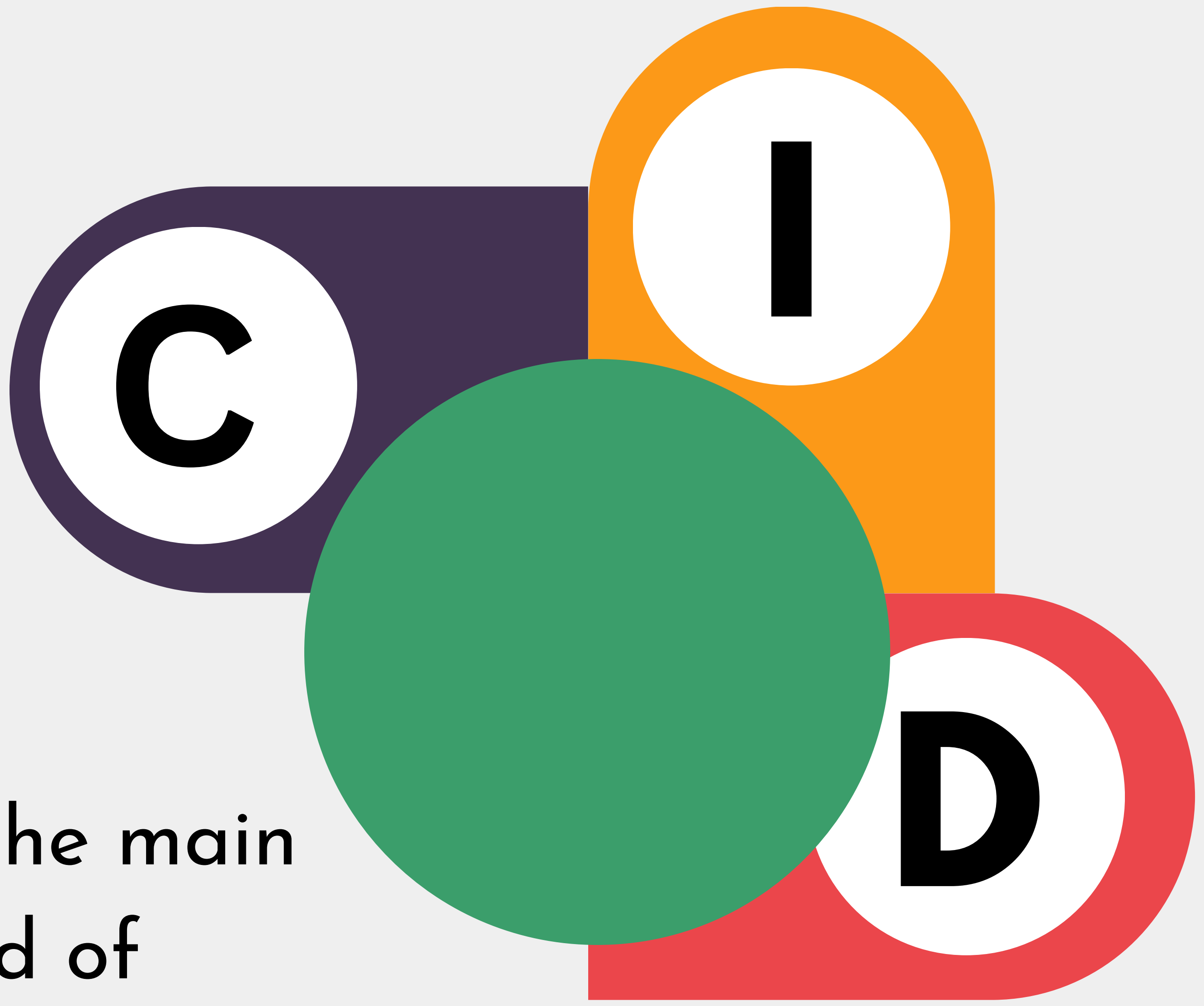
Stacked model accuracy



Real-time analysis or periodic analysis properties

1. SGDClassifier for further incremental learning need-based
2. XGBoost equipped with similar properties as well
3. Anomaly detection need-based

Column conundrum



Imbalance

Data Quality

Our CID agent will solve the main problems of every kind of financial data you feed it.

End

Thank you and we hope this helps. 🧁