Supervised Learning Project

Hilda Monterrubio
Monday, November 6th

Agenda

01 EDA process

02 Models (DTC & Ada)

03 Results

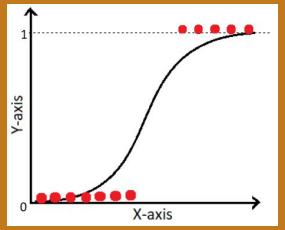
04 Conclusion

EDA process

1. Checked for duplicate values and null values (none found).
2. Ensured that each column had the appropriate data type.
3. Re-named columns appropriately using snake code.
4. Identified possible categorical columns (3 columns)).
5. Dropped low variance columns (liability assets flag & net income flag columns).
6. Splitted X & y.
7. Ran correlation, excluded 15 features that had a correlation greater than the defined threshold (**0.9** in this case).
8. Ran train-test split.
9. Executed scaling.
10. Handled imbalance data (mix of both, oversampling & undersampling to end up with a nice balance). Ended up with {0: 1123, 1: 749}.
11. Proceed to build the models based on the **resampled data.**

Models

Logistic Regression

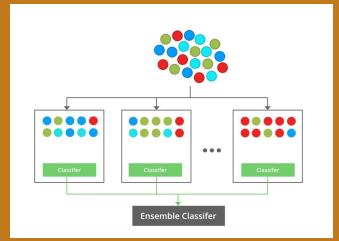


- Accuracy: 0.84%
- Recall: 0.17%
- Precision: 0.04%
- F1 Score: 0.06%

Hyperparameters tuning

- `Best parameters: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}`

XG Boost Classifier



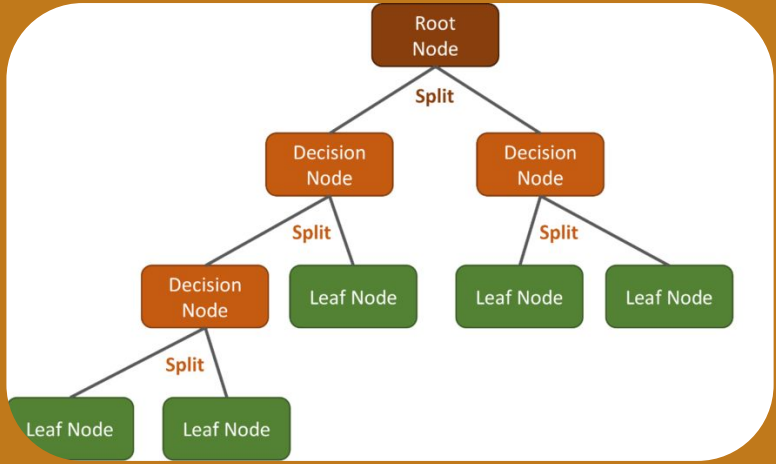
- Accuracy: 0.94%
- Recall: 0.58%
- Precision: 0.27%
- F1 Score: 0.36%

Hyperparameters tuning

- `Best Parameters: {'learning_rate': 0.5, ' max_depth': 3, 'n_estimators': 100, 'subsample': 0.8}`

Decision Tree Classifier

A model that constructs a tree to make decisions, where each node represents a "test" on a feature, the branch represents a decision rule, and each leaf node represents the outcome.



How does it work?

1. **Evaluation** - Before anything, it looks at all the possible questions it can ask based on the data features by calculating the **Gini impurity.**
2. **Splitting**- It chooses the **best question/feature** that splits the data into two meaningful groups.
3. **Follows the answers** - Based on the answer, it follows the path down the tree to the next question.
4. **Asks more questions** - Each group created by a question can be split further with more questions. Each time a new question is asked, you're getting closer to the answer.
5. **Decision making-** Eventually, there will be a small enough group to make a decision. This means you've assigned a class (e.g like "Play Outside" or "Stay In" for a weekend activity).

Decision Tree Classifier

Parameters

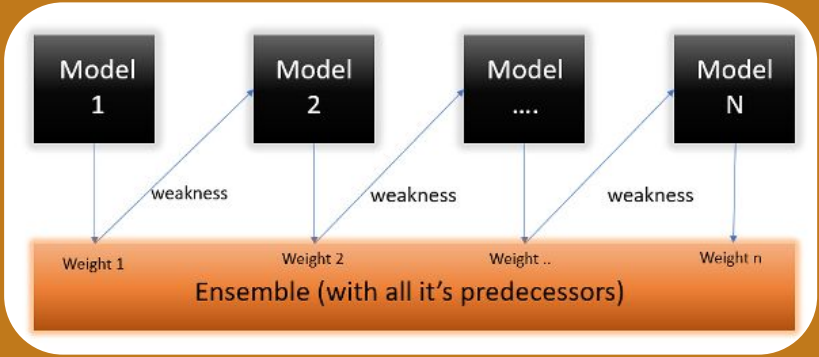
- **`max_depth`:** The maximum depth of the tree.
- **`min_samples_split`:** The minimum number of samples required to split an internal node.
- **`min_samples_leaf`:** The minimum number of samples required to be at a leaf node.
- **`criterion`:** The function to measure the quality of a split. Sklearn supports “gini” for the Gini impurity and “entropy” for the information gain.

From code

```
param_grid = {
'max_depth': [None, 10, 20, 30,
40, 50],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'criterion': ['gini', 'entropy']
}
```

AdaBoost Classifier

A model that combines multiple weaker models to form a strong classifier. It improves its accuracy by learning from the mistakes of previous models in a sequential manner, and then it combines them in a weighted manner to produce the final prediction.



How does it work?

1. **Trains weak learners -** It starts by training a weak learner (a simple decision tree, for instance) on the dataset and tries to classify it correctly.
2. **Adapts to mistakes -** The algorithm increases the emphasis on the data that was misclassified, and assigns a weight to errors. This process continues, with each new algorithm focusing on the errors of the previous one.
3. **Combines learners -** Since each of these algorithms might not be very smart on its own, AdaBoost combines them, and it gives more importance to the algorithms that perform the best.
4. **Creates a strong classifier -** The end result is a cumulative model that's made up of many weak learners, each compensating for the others' mistakes. The decision made by this strong learner is the one that is output by the model.

AdaBoost Classifier

Parameters

- **`n_estimators:`** The maximum number of estimators (weak learners) at which boosting is terminated.
- **`learning_rate:`** A weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier.
- **`random_state:`** Controls the randomness of the algorithm.

From code

```
param_grid = {
    'n_estimators': [10, 50, 100,
200],
    'learning_rate': [0.001,
0.01, 0.1, 1.0],
    'random_state’=42
}
```

Comparison table

With default values

Top 3

	Logistic Regression	1 XG Boost Classifier	Decision Tree Classifier	2 Gradient Boosting Classifier	3 AdaBoost Classifier
Accuracy	0.84	0.94	0.90	0.92	0.92
Recall	0.17	0.58	0.54	0.65	0.54
Precision	0.04	0.27	0.16	0.22	0.20
F1 score	0.06	0.36	0.25	0.33	0.29

Comparison table

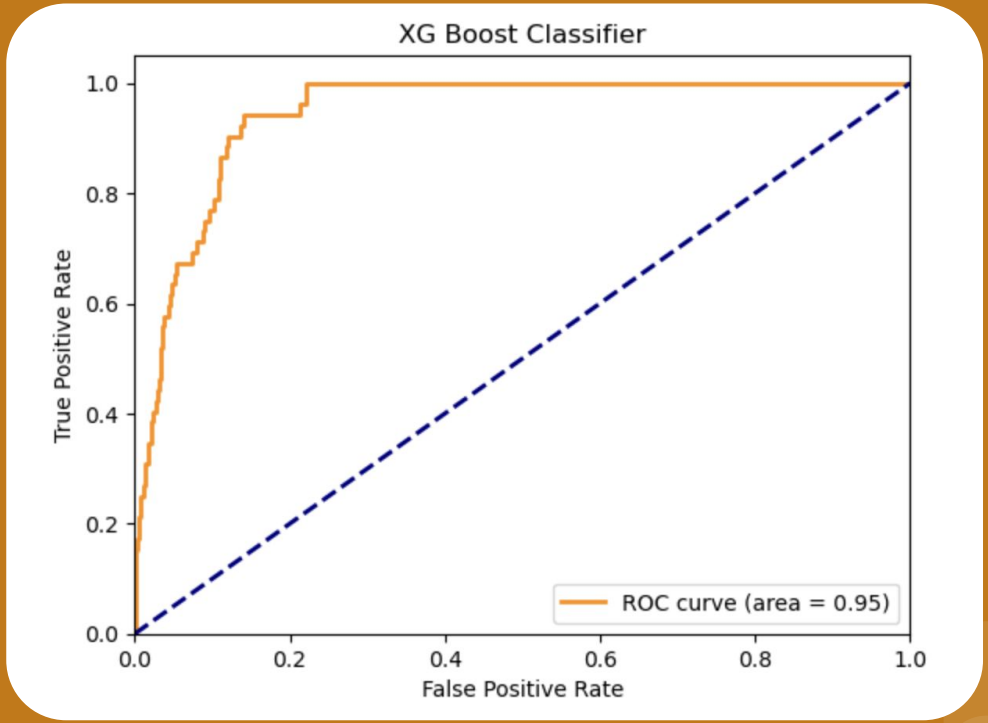
With Hyperparameter tuning

Top 3

	3 Logistic Regression	1 XG Boost Classifier	Decision Tree Classifier	2 Gradient Boosting Classifier	AdaBoost Classifier
Accuracy	0.91	0.94	0.90	0.94	0.92
Recall	0.73	0.62	0.52	0.54	0.54
Precision	0.21	0.30	0.16	0.25	0.20
F1 score	0.33	0.40	0.25	0.35	0.29
ROC score	0.92	0.95	0.72	0.90	0.88

ROC curve of the best model

XGBoost Classifier



Thank you!