Raw to Ready: A Practical Guide to Feature Engineering

Machine learning isn't just about throwing a bunch of data into an algorithm and expecting it to work. The truth is, the quality of your features matters as much—if not more—than the choice of your model. In this blog, we'll dive into the Bank Marketing Dataset, exploring how selecting the right set of features can dramatically impact a model's performance.

The dataset contains different features such as age and job with the target variable y, which indicates whether a customer subscribed to the bank or not. But here's the catch—not all features are equally helpful. Some carry meaningful information, while others add noise or redundancy, slowing down learning and hurting accuracy.

In this blog, we'll show you:

1. How feature selection influences model performance.
2. What happens when we use only the most or least correlated features.

Before diving into the details, I want to remind you some key concepts related to this experiment.

1. Correlation

Correlation helps identify relationships between features and the target variable.These correlations guided our feature selection process.

2. Accuracy

Accuracy measures the percentage of correct predictions. However, it's not always reliable for imbalanced datasets, so we also used precision, recall, and F1-score.

3. Logistic Regression

A simple and interpretable algorithm for binary classification. It uses a sigmoid function to calculate probabilities:

We tested Logistic Regression with two setups:

1. Most correlated features.
2. Least correlated features.

4. Confusion Matrix

The confusion matrix breaks down predictions into four categories:

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Predicted positive but actually negative (Type I error).
- False Negatives (FN): Predicted negative but actually positive (Type II error).

The Impact of Feature Selection

Feature selection improves:

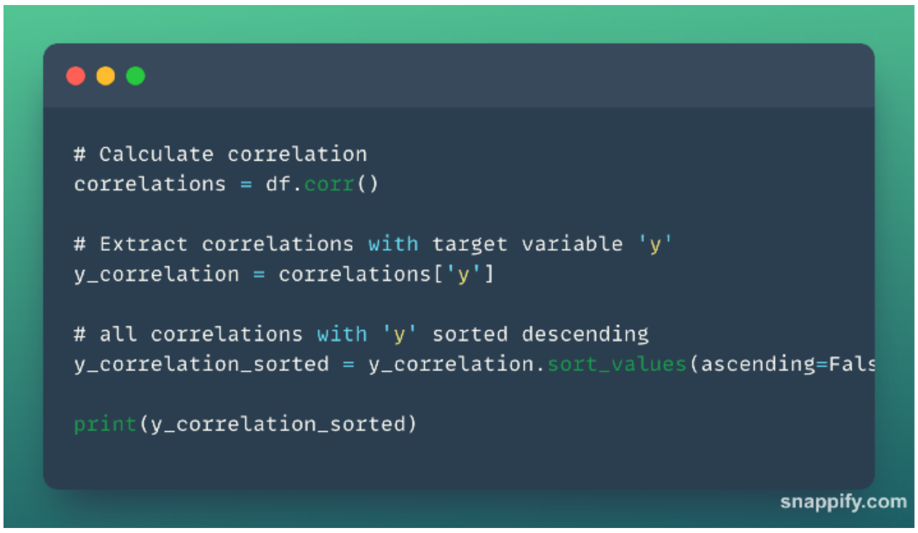
- Model simplicity by reducing noise and redundancy.
- Performance, but too few features can miss important patterns.

Now, Here are the steps for the experiment:

1- Calculate correlation of each feature with target variable y
2- Sort features as top correlations and least correlations
3- Create data with top 2 features
4- Train the model with Logistic Regression and evaluate the model
5- Create another data with the least 2 correlated features
6- train the model with those features and evaluate the model
7- compare the results

1. Calculate Correlation

We started by calculating the correlation between each feature and the target variable y. This helped us understand which features are most related to the prediction task.

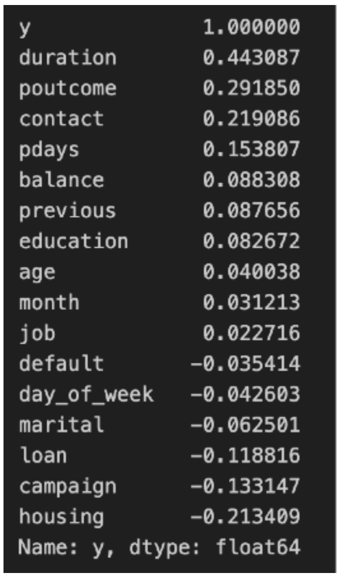


```python
# Calculate correlation
correlations = df.corr()

# Extract correlations with target variable 'y'
y_correlation = correlations['y']

# all correlations with 'y' sorted descending
y_correlation_sorted = y_correlation.sort_values(ascending=Fals

print(y_correlation_sorted)
```



```
y              1.000000
duration       0.443087
poutcome       0.291850
contact        0.219086
pdays          0.153807
balance        0.088308
previous       0.087656
education      0.082672
age            0.040038
month          0.031213
job            0.022716
default       -0.035414
day_of_week   -0.042603
marital       -0.062501
loan          -0.118816
campaign      -0.133147
housing       -0.213409
Name: y, dtype: float64
```

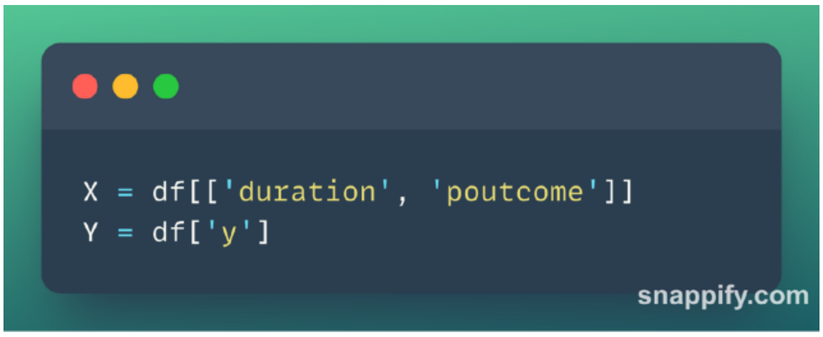
2. Sort Features by Correlation

This sorting allowed us to select meaningful features for training and exclude irrelevant ones.

Here are the results. The most correlated features are ‘duration' and ‘poutcome', the least correlated features are ‘age' and ‘month'.

3. Create Data with Top 2 Features

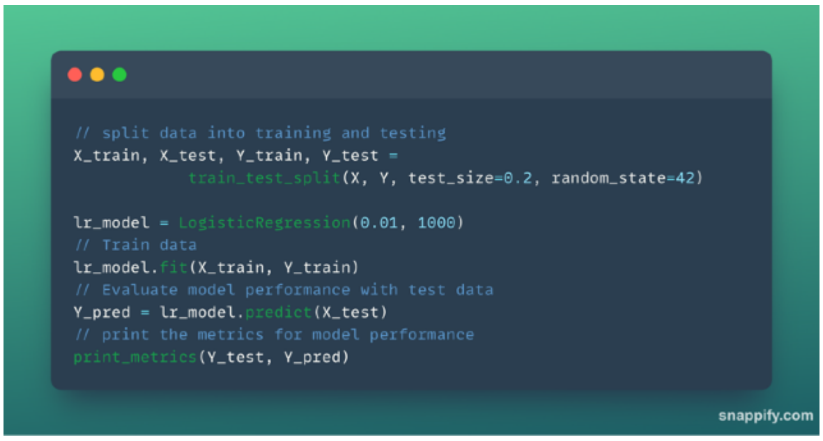
Using the two most correlated features, ‘duration’ and ‘poutcome’, we created a smaller dataset. This reduced feature set simplified the model while focusing on the most informative variables.



```
X = df[['duration', 'poutcome']]
Y = df['y']
```

4. Train the Model with Logistic Regression and Evaluate

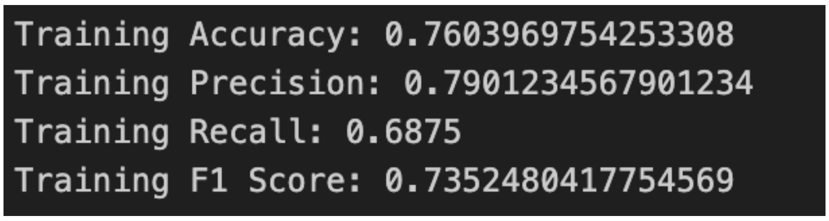
We trained a custom Logistic Regression model using the reduced dataset (only the top 2 features). After training, we evaluated the model on the test set.



```
// split data into training and testing
X_train, X_test, Y_train, Y_test =
            train_test_split(X, Y, test_size=0.2, random_state=42)

lr_model = LogisticRegression(0.01, 1000)
// Train data
lr_model.fit(X_train, Y_train)
// Evaluate model performance with test data
Y_pred = lr_model.predict(X_test)
// print the metrics for model performance
print_metrics(Y_test, Y_pred)
```

This step demonstrated that even with just two features, the model could achieve decent performance, validating the importance of feature selection. Let me know if you’d like any additional visuals or refinements!



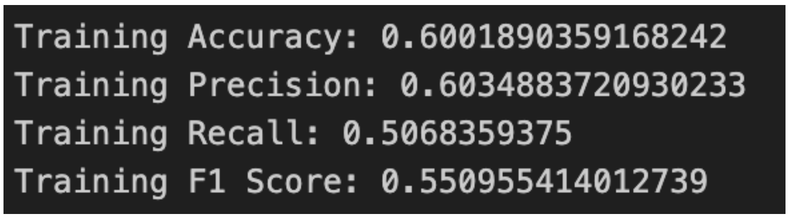
```
Training Accuracy: 0.7603969754253308
Training Precision: 0.7901234567901234
Training Recall: 0.6875
Training F1 Score: 0.7352480417754569
```

5. Create Data with the Least 2 Correlated Features

Next, we created a dataset using the two least correlated features, month (0.03) and job (0.02). These features have almost no relationship with the target variable y.

6. Train the Model with the Least Correlated Features and Evaluate

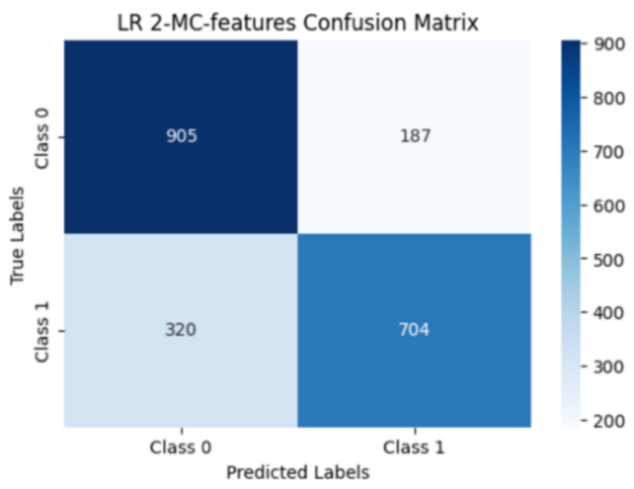
We trained the Logistic Regression model on this new dataset with only the least correlated features.



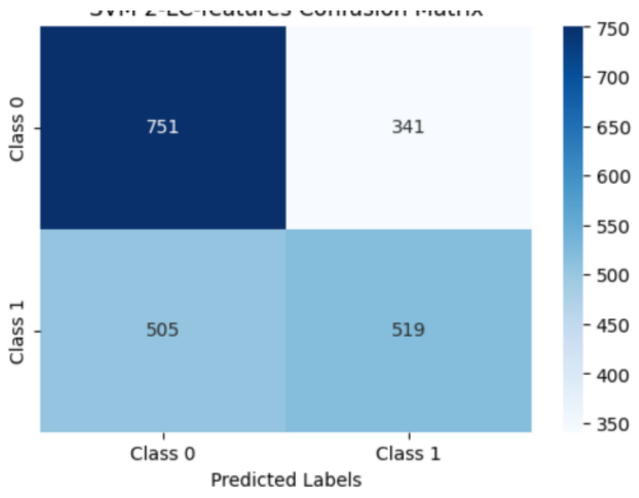
```
Training Accuracy: 0.6001890359168242
Training Precision: 0.6034883720930233
Training Recall: 0.5068359375
Training F1 Score: 0.550955414012739
```

7. Compare the Results

Finally, we compared the performance of the model trained on the most correlated features (duration and poutcome) versus the least correlated features (month and job).



Most Correlated Features Confusion Matrix



Least Correlated Features Confusion Matrix

As you can see, The model performs better with the more correlated features.

Conclusion

This experiment highlights the critical role of feature selection in machine learning. By comparing models trained on the most correlated features (duration and poutcome) versus the least correlated ones (month and job), we saw a stark difference in performance. Models with meaningful features achieved higher accuracy, precision, recall, and F1-scores, while irrelevant features significantly degraded performance.