Feature Importance

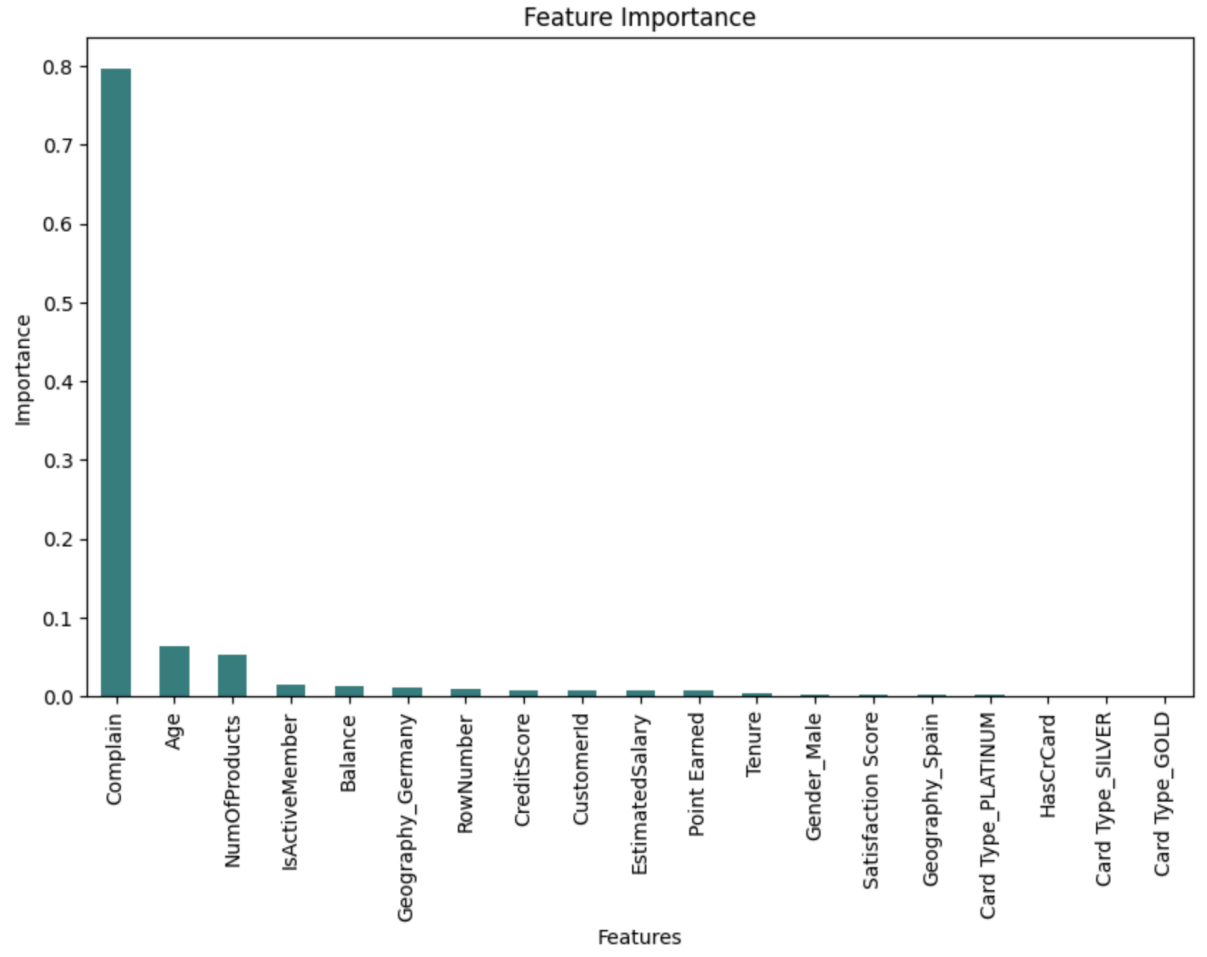
The feature importance plot from the Random Forest Classifier helps us understand which features have the most influence in predicting whether a customer exited or not.

The bar graph tells us:

1. Ranking of the features by importance:
   Features at the top of the bar chart contribute the most to the model's predictions.

2. Insights for Business Decision-Making
   Features like "Credit Score," "Age," or "Balance" have high importance comparably, which suggests these factors strongly influence customer churn. Features like "Geography" or "Gender" have low importance, it means they don't significantly impact churn, and we might reconsider their necessity.

3. Feature Selection
   High-importance features should be prioritized in model tuning, and low-importance ones can potentially be dropped to reduce model complexity.



Data Partitioning Strategy

I have used Stratified Sampling to partition the dataset into training, validation, and test sets. The goal is to ensure that each subset maintains the same proportion of the target variable (Exited) as the original dataset. This is particularly important when dealing with imbalanced classes, as it prevents bias in model training and evaluation.

Strategy:

1. Excluded the non-predictive columns such as RowNumber, and CustomerID as they don't contribute anything to the prediction.
   Excluded Exited since it is our target variable.

2. First Split (60% Train, 40% Temp):
   I used StratifiedShuffleSplit to ensure Exited proportions were preserved.

3. Second Split (50/50 Validation-Test from Temp):
   Split the temporary set (40%) equally into validation (20%) and test (20%). Again, I used StratisfiedShuffleSplit to maintain class balance.

I used StratisfiedShuffleSplit because:

1. Prevents Class Imbalance Issues: Ensures the distribution of Exited is similar across all sets.
2. Improves Model Performance: The model sees a representative mix of both classes during training.
3. Reliable Evaluation: Validation and test sets mimic the real-world distribution, leading to more generalizable results.

Performance of each trained Model

The results show the performance of various machine learning models trained on the given dataset.

1. Logistic Regression:
   Accuracy: 0.85, Precision: 0.71, Recall: 0.47
2. K-Nearest Neighbors:
   Accuracy: 0.75, Precision: 0.21, Recall: 0.08
3. Decision Tree:
   Accuracy: 1.00, Precision: 1.00, Recall: 0.98\
4. Support Vector Machine:
   Accuracy: 0.80, Precision: 0.00, Recall: 0.00
5. Random Forest:
   Accuracy: 1.00, Precision: 1.00, Recall: 1.00
6. XGBoost:
   Accuracy: 1.00, Precision: 1.00, Recall: 1.00
7. Gradient Boosting:
   Accuracy: 1.00, Precision: 1.00, Recall: 1.00
8. AdaBoost:
   Accuracy: 1.00, Precision: 1.00, Recall: 1.00
9. LightGBM:
   Accuracy: 1.00, Precision: 1.00, Recall: 1.00

Many models such as Decision Trees, Random Forest, XGBoost, and more depicted perfect accuracy, precision, and recall. However, this might indicate that either the data is not enough or simple enough for these complex models to overfit.

The general rule of thumb is implementing the model that delivers the performance and is the simplest of all. With the results I have, Logistic Regression can be used here as it is the simplest model with fine performance and not an overfitted model accuracy.

If a much larger dataset comes, I would recommend either Random Forest or XGBoost as they have higher accuracy while being relatively interpretable. Random Forest is also more transparent with its functions. They have general robustness and resistance to overfitting and are computationally efficient for larger datasets.

After Feature Importance, since Complain seemed to be highly relevant and was the reason for overfitting, after removing it, the model's performance is as follows:

1. Decision Tree:
   Accuracy: 0.77, Precision: 0.45, Recall: 0.59
2. K-Nearest Neighbors:
   Accuracy: 0.81, Precision: 0.53, Recall: 0.36
3. Logistic Regression:
   Accuracy: 0.81, Precision: 0.56, Recall: 0.30
4. Support Vector Machine:
   Accuracy: 0.86, Precision: 0.76, Recall: 0.45
5. Random Forest:
   Accuracy: 0.85, Precision: 0.65, Recall: 0.56
6. XGBoost:
   Accuracy: 0.83, Precision: 0.60, Recall: 0.52
7. Gradient Boosting:
   Accuracy: 0.85, Precision: 0.66, Recall: 0.56
8. AdaBoost:
   Accuracy: 0.84, Precision: 0.62, Recall: 0.54
9. LightGBM:
   Accuracy: 0.85, Precision: 0.67, Recall: 0.55

Now, SVM also seems to be a great choice as a model to choose as it has one of the highest accuracies and the highest precision rate, that is has the lowest false positive rate, which in this case is required.
Given the problem statement, we require a balanced and strong performant across all metrics, making SVMs the highly likable choice here.

<u>Improvements:</u>

1. Hyperparameter Tuning: Apply grid search or Bayesian optimization for models like Random Forst and XGBoost.
2. Validation Metrics: Consider AUC-ROC, F!-Score, and confusion matrix analysis instead of just accuracy.
3. Regularization: Add L1/L2 regularization for Logistic Regression, Decision Trees, and ensemble models.