Problem Statement:
The company aims to create an automated recruitment system that assists in identifying strong candidates for the sales officer role during the application process.

Feature Engineering:

The dataset consists of 22 columns

Regarding the 'Residential Pincode' and 'Branch Pincode,' I attempted to find an API capable of calculating **the distance between two pincodes** but was unsuccessful. Instead, I **utilized the first three digits of the pincode**, associating them with their respective states. This approach aims to **prioritize candidates from the same state**, presuming they possess a **better understanding of the local language**, which is vital for effective sales communication and building trust. As we know for each state pincode is in some range like:-
Source link:-
https://docs.cleartax.in/cleartax-docs/e-invoicing-api/e-invoicing-api-reference/resources-and-master/pincode-state-mapping-pattern

Ex:- Like ex:- "WEST BENGAL": [700, 743],
"JHARKHAND": [813, 835],
"ODISHA": [751, 770],
So I interchange this pincode with the first three-digit with the respective states

The column **'How many organizations have you worked for previously?'** contained erroneous data such as '1-Feb.' To rectify this, I interpreted 'Feb' as 2 and 'May' as 5, converting entries like '1-Feb' to '1-2' or '3-May' to '3-5.' Additionally, I removed entries containing 'Feb' and 'May' from the dataset, resulting in a slight increase in model accuracy.
Ex:- 1-Feb, 3-May as 1-2, 3-5

I parsed the **'Date of Joining'** into three separate columns for day, month, and year. This facilitates better comprehension for machine learning algorithms, as directly encoding the year can introduce complexity due to messy data.

In the **'How many earning members are in your family?'** column, there were unrealistic values like '10,000' or '20,000.' I replaced such values with either '10' or the mean/median of the entire column.

Removing the '**Name of your Previous Organization / Company**' column improved model accuracy, as it contained excessive data that did not significantly differentiate between candidates.

The '**Number of Family Members**' column was removed from consideration, as it does not impact a salesperson's performance and thus does not contribute to the model's predictive power.

I looked at graphs for some **columns with small values** to see if they were related to performance. However, I didn't find anything interesting or important from the analysis. It seems like these factors might not be very good at predicting performance.

'Products Sold in Previous Role' and 'Department' provided redundant information. Consequently, I retained the 'Department' column, as it indicates the types of products the candidate is interested in selling.

Dividing **'Average Ticket Size Handled in Previous Role'** by **'Incentive Earned in Previous Role'** yielded valuable insights into the salesperson's performance.

For missing or NaN values, I utilized **mean imputation** as the strategy, employing the SimpleImputer from the sklearn library. After experimenting with different imputation strategies, **median imputation** proved to be the most effective, resulting in improved model accuracy. Unusually large values in certain columns, such as 'Family Size,' were replaced with either the median or a random value within a plausible range.

Models Used:

I employed various models, including:

Random Forest
Naive Bayes
Logistic Regression
Support Vector Machine
Neural Network
CatBoost
K-Nearest Neighbors

Fine-tuning of hyperparameters
While fine-tuning the hyperparameters of the Neural Network, I experimented with parameters such as:
Batch size
Learning Rate
Optimizer
Activation function

Although I fine-tuned the **hyperparameters of the Neural Network**, achieving accuracy between 55% to 60%, it was comparable to or slightly lower than Random Forest, which emerged as the best-performing model.

In CatBoost, despite experimenting with different **hyperparameters** such as **learning rate**, **number of trees**, and **maximum depth**, the accuracy remained consistent with that of Random Forest.
The best-performing hyperparameters for catboost were
:- **iterations=1000, learning_rate=0.1, and depth=6.**

Factors Affecting Model Accuracy:

Possible missing features
Errors in the dataset due to incorrect values
Missing values and dummy CVs containing no useful information

Training and Testing:

For training and testing, I utilized separate train and test datasets. However, for evaluating the test data, I trained the entire model on the training dataset and then predicted the output for the test dataset. **Given the dataset's size, training the model on the entire dataset was more efficient than using a subset.**

Conclusion:

The Random Forest classifier emerged as the best model for this problem statement.