TERM PROJECT – FINAL REPORT

ATTRITION PREDICTION

DOMAIN

HR ANALYTICS

SUBMITTED BY GROUP 7:

Albin Babu Varghese
Ali Guneysel
Erinda Kapllani
Jai Vigneshwar Aiyyappan
Kaushal Ghelani
Kumud
Madhoumithaa Veerasethu
Rajvi Patel

AASD 4001 MATHEMATICAL CONCEPTS FOR MACHINE LEARNING

CONTENT

Contents

Introduction:

In the ever-evolving job market, employee retention stands as a paramount concern for organizations. The consequences of high employee turnover encompass increased recruitment expenses, the loss of invaluable talent, and disruptions in operational continuity. Predicting an employee's inclination to stay or leave is a pivotal insight that empowers organizations to adopt preemptive measures for talent retention.

This report delves into the application of machine learning techniques to confront the challenge of employee retention. By harnessing historical employee data, our objective is to construct a predictive model capable of reliably gauging the probability of an employee departing from the company. This predictive model emerges as a potent tool for HR departments and organizational leadership, facilitating the implementation of focused retention strategies aimed at mitigating turnover rates.

Dataset Source: Kaggle (https://www.kaggle.com/datasets/tawfikelmetwally/employee-dataset)

GitHub Repo of the project:

Problem Statement:

The problem involves predicting employee attrition or, in simpler terms, determining if an employee is likely to leave the company soon. To address this, we will use a machine learning approach to analyze a dataset containing various employee-related features, such as their education level, joining year, city of residence, payment tier, age, gender, previous benching experiences, years of experience in their current domain, and whether they eventually left the organization or not.

The primary objective of this project is to build a predictive model that can accurately classify employees into two categories: those likely to stay and those likely to leave. By doing so, we aim to provide HR departments and organizational leaders with a tool that can assist in the early identification of employees at risk of leaving, enabling them to take proactive measures to improve employee retention.

In this report, we will detail the steps of the machine learning process, including data preprocessing, feature selection, model selection, evaluation metrics, and insights gained from the analysis. Additionally, we will discuss the implications of our predictive model and how it can be effectively integrated into HR practices to create a more engaged and stable workforce.

EDA:

Exploratory analysis of the employee dataset involves assessing relationships among variables and understanding the distribution of employees based on several factors. Using this dataset, we have used several techniques to understand how our predictive model can be used effectively in HR practices.

1. **Correlation Analysis:**

- The correlation matrix is calculated to quantify the relationships between numeric variables

- A heatmap is generated to visually represent the correlations using the seaborn library.
- This aids in identifying potential patterns and dependencies among numerical features.

2. **Cluster Analysis:**

- KMeans clustering is applied to the 'JoiningYear' and 'Age' features, with an additional consideration for whether an employee left or not ('LeaveOrNot').
- The scatterplot visualizes how employees are distributed in the 'JoiningYear' vs. 'Age' space, with distinct colors indicating whether they left the company or not.
- Clustering helps identify groups of employees with similar characteristics.

3. **Label Encoding for Categorical Data:**

- Categorical variables such as 'Education,' 'City,' 'Gender,' and 'EverBenched' are label-encoded using sklearn's LabelEncoder.
- Label encoding transforms categorical data into numerical form, facilitating machine learning model compatibility.

4. **Correlation Matrix for Label Encoded Data:**

- A correlation matrix is generated for the label-encoded dataset to observe relationships among variables after encoding.
- The heatmap visualizes correlations, providing insights into the relationships between encoded categorical features and other variables.

5. **Feature Separation:**

- Features ('X') and the target variable ('y') are separated for modeling purposes.
- Numerical and categorical column names are identified, which is crucial for subsequent preprocessing and modeling.

6. **Column Lists:**

- Lists of numerical and categorical columns are printed for reference.
- This step aids in further analysis, preprocessing, and selecting appropriate features for modeling.

In summary, the exploratory analysis includes correlation visualization, cluster analysis, label encoding for categorical variables, and the separation of features and target variables. These steps provide a comprehensive understanding of the dataset and lay the work for other machine learning tasks.

Statistical tests:

It is necessary to conduct statistical tests to assess the significance of features in relation to the target variable. The analysis involves two types of tests: Analysis of Variance (ANOVA) for numerical columns and Chi-Square tests for categorical columns.

a. ANOVA for Numerical Columns:

a. For each numerical column (e.g., JoiningYear, PaymentTier, Age, ExperienceInCurrentDomain), ANOVA is performed.
b. The p-value from the ANOVA test is printed, and based on the threshold of 0.05, features are classified as either "Significant" or "Non-significant."
c. In this case, all numerical features result to be significant as their p-values are less than 0.05.

b. Chi-Square Tests for Categorical Columns:

a. For each categorical column (e.g., Education, City, Gender, EverBenched), a Chi-Square test is conducted to evaluate the association between the categories and the target variable.
b. The Chi2 statistic and p-value from the test are printed, and again, features are classified as "Significant" or "Non-significant" based on the 0.05 threshold.
c. All categorical features are significant, indicating they have a statistically significant association with the target variable.

Data Preprocessing Procedure:

Data preprocessing is a crucial step in preparing the dataset for machine learning. It involves several operations to ensure the data is clean, consistent, and ready for model training. Even though the given dataset does not contain any null values, we have implemented data preprocessing steps that can handle such scenarios in real data. Below is the procedure for data preprocessing:

1. Handling Null Values:

1.1. While the current dataset does not contain any missing values, we have proactively included an imputer in the preprocessing pipeline to handle potential null values that may arise in real-world data.
1.2. The SimpleImputer is set to use the 'mean' strategy for numerical features and the 'most_frequent' strategy for categorical features. This ensures that missing values are replaced with the mean for numerical features and the most frequent category for categorical features.

2. Scaling Numerical Features:

2.1. Numerical features often have different scales, which can affect the performance of machine learning algorithms. To address this, we apply the StandardScaler to standardize the numerical features.
2.2. Standardization scales the features to have a mean of 0 and a standard deviation of 1, making them more suitable for modeling.

3. One-Hot Encoding Categorical Features:

3.1. Categorical features, such as 'Education,' 'City,' and 'Gender,' need to be transformed into a numerical format for machine learning algorithms to work effectively.
3.2. We use the OneHotEncoder to convert categorical features into binary vectors (one-hot encoding) while handling unknown categories with the 'ignore' strategy.
3.3. Additionally, we use the 'drop' parameter set to 'if_binary' to drop one of the binary columns created for binary categorical features, reducing multicollinearity.

4. **Column Transformation:**
    4.1. The ColumnTransformer combines the preprocessing steps for numerical and categorical features.
    4.2. It applies the numerical_transformer to numerical columns and the categorical_transformer to categorical columns.
5. **Data Splitting:**
    5.1. The dataset is divided into training and testing sets using the train_test_split function.
    5.2. We allocate 80% of the data for training and 20% for testing.
    5.3. The stratify parameter is set to y to ensure that the class distribution is preserved in both the training and testing sets, given the imbalanced class distribution in the dataset.
6. **Creating the Full Pipeline:**
    6.1. The preprocessing steps are integrated into a full data preprocessing pipeline using the Pipeline class.
    6.2. The main component of this pipeline is the preprocessor, which combines all the transformations and imputations.

By following these data preprocessing steps, we ensure that the dataset is ready for model training and evaluation. The inclusion of the imputer makes the pipeline robust to missing data, which can be beneficial for real-world scenarios where data quality may vary. These preprocessing steps are essential for building a reliable predictive model for employee retention.

Logistic Regression:

Definition:

Logistic Regression is a statistical model used in machine learning for classification tasks. It is a predictive analysis technique used to describe data and explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables.

Process:

- Created a simple logistic regression model using the provided dataset.
- Utilized GridSearchCV to explore various parameter combinations and identify the optimal hyperparameters, which turned out to be 'C=1', 'penalty='l1', and 'solver='saga'.
- Extracted the coefficients of the features from a trained Logistic Regression model and visualized their significance.
- Removed features with importance values less than 0.1 and then conducted another GridSearchCV with the previously mentioned parameters. In this case, the best parameters were found to be 'C=100', 'max_iter=1000', 'penalty='l1'', and 'solver='liblinear''.
- Finally, we constructed a final model using these best parameters, ensuring that it yielded the best scoring metrics.

Model Performance:

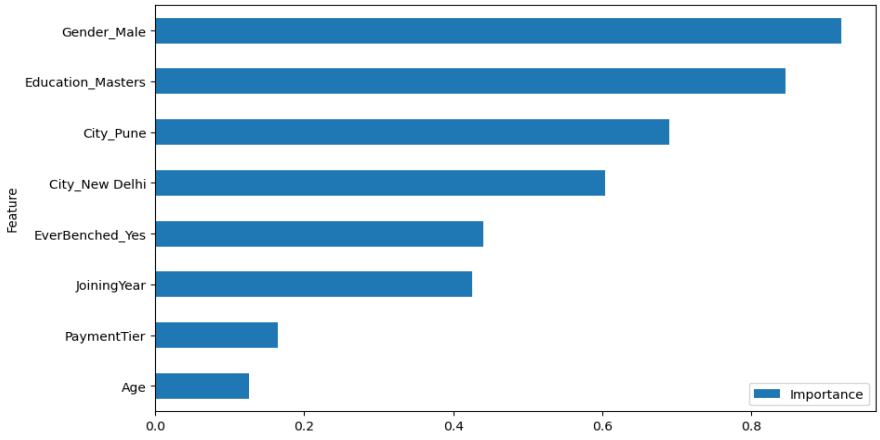
This table provides a comprehensive overview of three distinct machine learning models, each characterized by unique hyperparameters and sets of features. The assessment of these models encompasses a range of performance metrics, including precision, recall, AUC (Area Under the Curve), F1 score, accuracy, and a detailed examination of their confusion matrices.

Description	Parameters	No of Features	Precision	Recall	AUC	F1 Score	Accuracy	Confusion Matrix
Base model	-	12	0.67	0.42	0.66	0.52	0.73	[[543,68], [184,136]]
Grid Search with all Features	{'C': 1, 'penalty': 'l1', 'solver': 'saga'}	12	0.67	0.42	0.66	0.52	0.73	[544, 67], [184,136]
Grid Search with reduced features	**{'C': 100, 'max_iter': 1000, 'penalty': 'l1', 'solver': 'liblinear'}**	**8**	**0.68**	**0.43**	**0.66**	**0.53**	**0.74**	**[[547,64], [181,139]]**

Discussion:

Among the models evaluated in the table, the one generated using GridSearchCV with 8 features demonstrates slightly superior performance across all the scoring metrics listed. However, it is worth noting that the enhancements observed were minor.

Conclusion:



The model utilizing 8 features along with the hyperparameters {'C': 100, 'max_iter': 1000, 'penalty': 'l1', 'solver': 'liblinear'} achieved the top-performing results.

Decision Tree Classifier:

Definition:

A decision tree classifier is a machine learning algorithm that makes predictions by recursively splitting a dataset into subsets based on the values of its input features, creating a tree-like structure of decision rules. It is used for classification tasks, where it assigns a label or category to a data point based on the path through the tree that leads to that point.

Process:

- Created a baseline Decision Tree Classifier.
- Used GridSearchCV with a combination of parameters {"criterion": ["gini", "entropy"], "max_depth": [2,4,10],"max_features": [None,'auto','sqrt','log2',5]} to find the best combination of hyper parameters.
- Used Pearson Correlation Function to find least number of features with performing high accuracy score. Used only three features ([“City_Pune,” “JoiningYear,” “Gender_Male”] and applied GridSearchCV with same previous parameters.
- Used AUC scores to find the least number of features with performing high AUC score. The reason is to keep the AUC score as high as possible since our dataset is unbalanced.
- Created a final model with the best scoring metrics

Model Performance:

This table presents a comparison of the four machine learning models. The models differ in terms of their hyperparameters and feature sets. They are evaluated across various metrics like precision, recall, AUC, F1 score, accuracy, and confusion matrix.

Description	**Parameters**	**No of Features**	**Precision**	**Recall**	**AUC**	**F1 Score**	**Accuracy**	**Confusion Matrix**
Base model	criterion = gini, max_depth = None, max_features = None	12	0.78	0.72	0.80	0.75	0.83	[546, 65] [91, 229]
Grid Search with all Features	criterion = entropy, max_depth = 10, max_features = None	12	0.84	0.71	0.82	0.77	0.85	[569, 42] [92, 227]
Grid Search with Reduced features by Accuracy (3)	criterion = gini, max_depth = 10, max_features = None	3	0.89	0.54	0.75	0.67	0.82	[589, 22] [148,172]

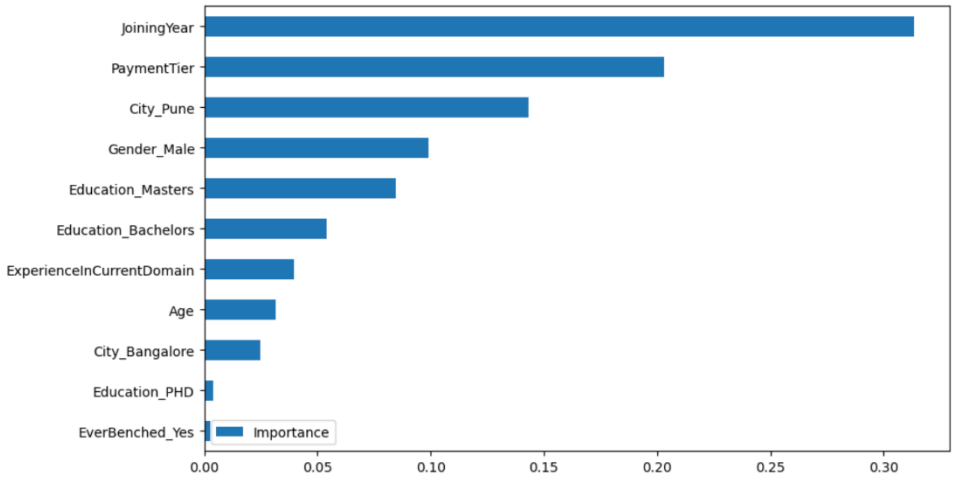
Grid Search with reduced features by AUC score (11)	**criterion = entropy, max_depth = 6, max_features = 11**	**11**	**0.85**	**0.70**	**0.85**	**0.77**	**0.85**	**[570, 41] [95, 225]**

Discussion:

The model with 11 features predicted by GridSearchCV performs better than the other models in terms of all the score metrics mentioned in the table.

The Baseline model performed poorest at precision while the model with 3 features performed poorest result at recall score.

Conclusion:



Model with 11 features and parameters {criterion = entropy, max_depth = 6 ,max_features = 11} performed the best.

Support Vector Machine (SVM):

Definition:

A supervised machine learning algorithm that classifies data into classes by creating a hyperplane that maximizes the margin between classes. It is commonly used for both classification and regression problems. In this section of the report, we create and analyze the performance of multiple SVM models with varying parameters and features to answer our problem statement.

Process:

- Created a baseline SVM
- Used GridSearchCV with a combination of parameters {'kernel': ['linear',’poly,’'rbf'], 'C': [1,2,3,4,5], 'gamma': [0.1,0.3,0.7,0.9,1],’class_weight': ['balanced']} to find the best combination of hyper parameters
- Used correlation matrix to find the features that are least correlated with the target variable. Removed 5 features 'Age', 'ExperienceInCurrentDomain', 'Education_PHD', 'City_New Delhi', 'EverBenched_Yes' with correlation <0.1
- Used GridSearchCV on the dataset with removed features with the following hyperparameter s
  param_grid1 = {
  'kernel': ['linear','rbf'] ,
  'C': [1,2,3,4,5],
  'gamma': [0.1,0.3,0.5,0.7,1,1.2,1.4,1.6,1.8,2.0],
  'class_weight': ['balanced']
- Created a final model with the best scoring metrics

Model Performance:

This table presents a comparison of the three machine learning models. The models differ in terms of their hyperparameters and feature sets. They are evaluated across various metrics like precision, recall, AUC, F1 score, accuracy, and confusion matrix.

Description	**Parameters**	**No of Features**	**Precision**	**Recall**	**AUC**	**F1 Score**	**Accuracy**	**Confusion Matrix**
Base model	kernel=linear, C =1, class_weight= balanced	12	0.58	0.68	0.71	0.62	0.72	[453, 158], [104, 216]
Grid Search with all Features	**C = 2, class_weight = balanced, gamma = 0.3, kernel = rbf**	**12**	**0.81**	**0.76**	**0.83**	**0.78**	**0.85**	**[552, 59], [76, 244]**
Grid Search with reduced features	C = 1, class_weight = balanced, gamma = 0.3, kernel = rbf	7	0.71	0.78	0.81	0.75	0.82	[510, 101], [70, 250]

Discussion:

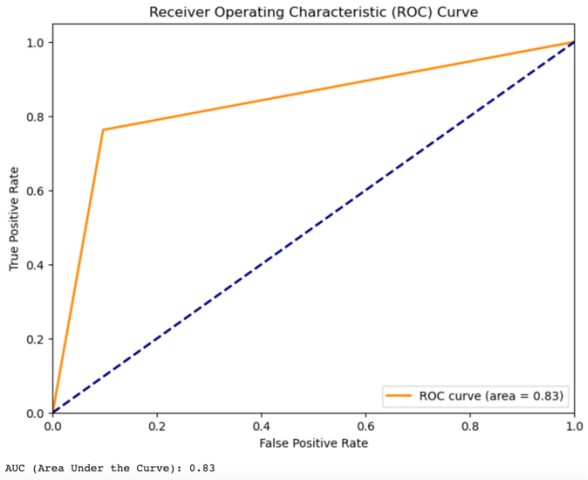
The model with all features predicted by GridSearchCV performs better than the other models in terms of all the score metrics mentioned in the table.

The Baseline model performs the poorest with low Precision, Recall and F1 score compared to other models

The model with reduced features predicted by GridSearchCV strikes a balance between the other two models offering a moderate performance with reduced features.

Conclusion:

Model with all 12 features and parameters C=2, kernel =’rbf’, class_weight= ’balanced’, gamma=0.3 performed the best.



Stochastic Gradient Descent (SGD) Classifier:

Definition:

SGD is an optimization algorithm used in machine learning to iteratively update model parameters by randomly selecting and using a single data point (or a small batch of data) to compute gradients, making it faster for large datasets.

Process:

The model chosen for this analysis is a Stochastic Gradient Descent (SGD) Classifier. The following steps were performed during model training:

1. Model Initialization: An SGD Classifier was initialized with parameters such as max_iter, Tol, and random_state.
2. Model Training: The SGD model was trained on the training dataset.

3. Hyperparameter Tuning: A grid search was conducted to find the best hyperparameters for the model, including 'loss,' 'alpha,' 'penalty,' and 'max_iter.'
4. Model Refinement: The model was refined with the best hyperparameters found during the grid search.
5. Feature Importance: Feature importance was analyzed, and less prominent features were dropped from the dataset.
6. Model Evaluation: The model's performance was evaluated using various metrics, including accuracy, precision, recall, AUC, and F1 Score.

Model Performance:

This table presents a comparison of the three machine learning models. The models differ in terms of their hyperparameters and feature sets. They are evaluated across various metrics like precision, recall, AUC, F1 score, accuracy, and confusion matrix.

Description	**Parameters**	**No of Features**	**Precision**	**Recall**	**AUC**	**F1 Score**	**Accuracy**	**Confusion Matrix**
Base model	**max_iter=1000, tol=1e-3, random_state=42**	**12**	**0.63**	**0.42**	**0.64**	**0.50**	**0.72**	**[534, 77], [187, 133]**
Grid Search with all Features	alpha=0.01, loss='log_loss', max_iter=500, random_state=42	12	0.68	0.36	0.64	0.47	0.72	[556, 55], [204, 116]
Grid Search with reduced features	alpha=0.01, loss='log_loss', max_iter=500 random_state=42	7	0.67	0.35	0.63	0.46	0.72	[557, 54], [209, 111]

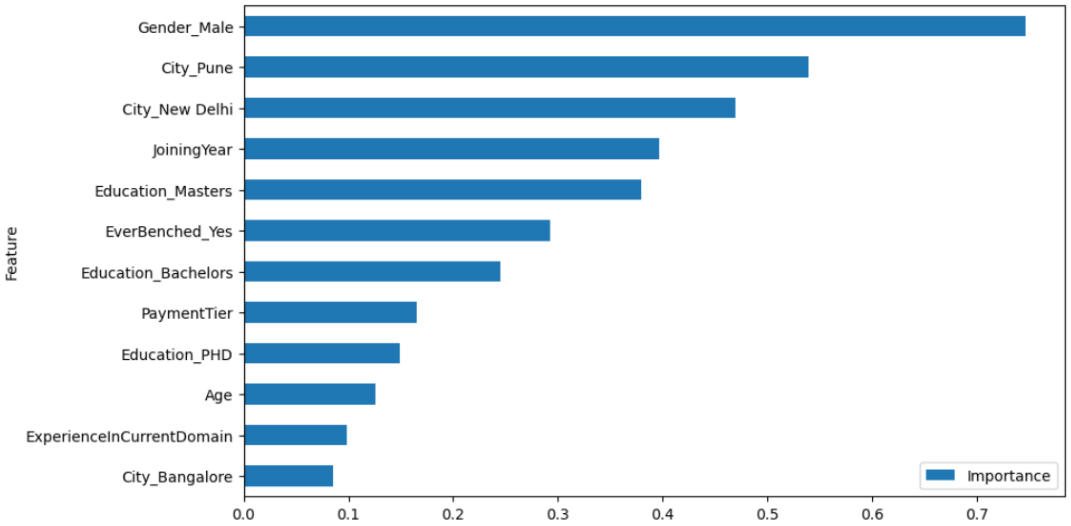
Discussion:

The base model with all features performs better than the other models in terms of AUC and recall metrics mentioned in the table.

The model with reduced features predicted by GridSearchCV performs the poorest with low Accuracy, Recall and Precision compared to other models.

The model with all features predicted by GridSearchCV performance a balance between the other two models offering a moderate performance with reduced features.

Conclusion:



The model utilizing 12 features along with the hyperparameters {alpha=0.001, penalty='l1', random_state=42} achieved the top-performing results.

Random Forest:

Definition:

Multiple decision trees are built in a random forest classification utilizing different random subsets of the data and characteristics. Each decision tree functions as an expert, advising on how to classify the data. Predictions are made by computing the prediction for each decision tree and then selecting the most popular outcome.

1. **Base Model:**
    a. Started by creating a base Random Forest Classifier (rfc_model) and fitted it to the training data.
    b. Predictions were made on the test data, and a confusion matrix was computed.
    c. Various metrics, including Precision, Recall, AUC, F1 Score, and Accuracy, were calculated for the base model.
2. **Grid Search on All Features:**
    a. A grid search (GridSearchCV) was conducted with different hyperparameters to optimize the Random Forest model.
    b. The grid search yielded the best hyperparameters (best_params_RFC) and the corresponding best score (best_score_RFC).
    c. The Random Forest model (rfc_model_grid) was updated with the best hyperparameters and fitted to the training data.
    d. Predictions were made using this optimized model, and a new confusion matrix and metrics were computed.

3. **Grid Search on Reduced Features:**
    a. Based on feature importance, less important features were identified and dropped from the training and test datasets.
    b. The grid search process was repeated to find the best hyperparameters for the reduced-feature model (rfc_model_grid_reduced).
    c. The model was updated with the best hyperparameters and fitted to the reduced-feature training data.
    d. Predictions were made with this model, and a new confusion matrix and metrics were computed.
4. **Final Model for Random Forest:**
    a. A final Random Forest model (rfc_model_final) was created, and its hyperparameters were set to the best parameters found during the grid search.
    b. The final model was fitted to the original training data.
    c. Predictions were made using this final model on the test data, and a confusion matrix and metrics were calculated.
    d. Feature importance was also determined for this final model.
5. **Recording Model Information:**
    a. Recorded essential information, including the model type (Random Forest Classifier), the number of features used, and the list of features, in a data structure named final_scores.
    b. These actions reflect a comprehensive approach to model selection, hyperparameter tuning, and feature selection, creating an optimized Random Forest Classifier model.
    c. Diligently documented key details in the final_scores data structure, demonstrating a thorough analysis and optimization process in machine learning.

Model Performance:

This table presents a comparison of the three machine learning models. The models differ in terms of their hyperparameters and feature sets. They are evaluated across various metrics like precision, recall, AUC, F1 score, accuracy, and confusion matrix

Description	Parameters	No of Features	Precision	Recall	AUC	F1 Score	Accuracy	Confusion Matrix
Base model		12	0.80	0.70	0.80	0.75	0.84	[[554,57], [96,224]]
Grid Search with all Features	**Bootstrap: True Criterion: Entropy Max Depth: 10 Max Features: Auto Number of Estimators: 1000**	**12**	**0.87**	**0.70**	**0.82**	**0.78**	**0.86**	**[[578,33], [96,224]]**
Grid Search with	Bootstrap: True Criterion: Gini Max Depth: 4	8	0.90	0.52	0.74	0.66	0.81	[[593,181], [155,165]]

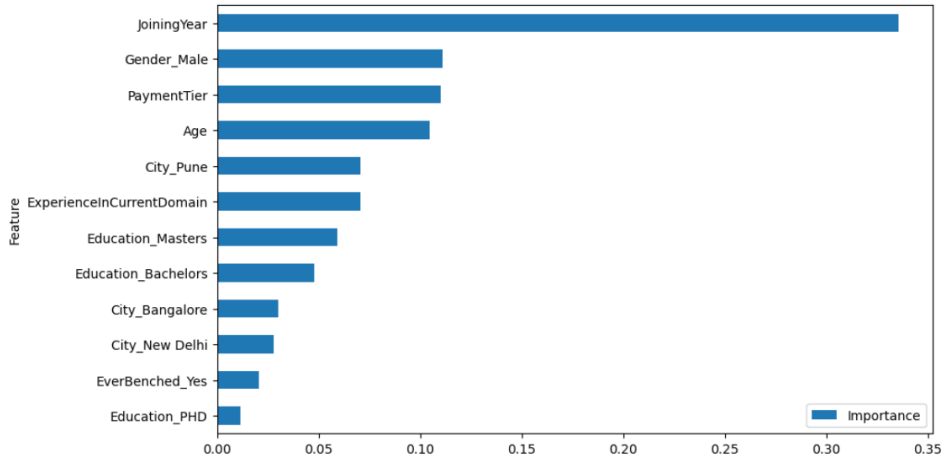
reduced features	Max Features: Auto Number of Estimators: 200							

Discussion:

The base model achieved an accuracy of 84% but had room for improvement in precision, recall, and AUC. Grid Search with all features resulted in better overall performance, with an accuracy of 86% and improved precision, recall, and AUC. Grid Search with reduced features sacrificed recall (52%) for high precision (90%), leading to a lower F1 score (66%) and moderate AUC (74%).

The Grid Search with all features demonstrated the best balance of performance metrics, offering enhanced accuracy and discrimination capability compared to the base model. The reduced-feature model excelled in precision but at the cost of recall and overall predictive power. Model choice should align with project priorities and trade-offs.

Conclusion:



The model utilizing 12 features along with the hyperparameters {Bootstrap: TRUE, Criterion: Entropy, Max Depth: 10, Max Features: Auto ,Number of Estimators: 1000} achieved the top-performing results.

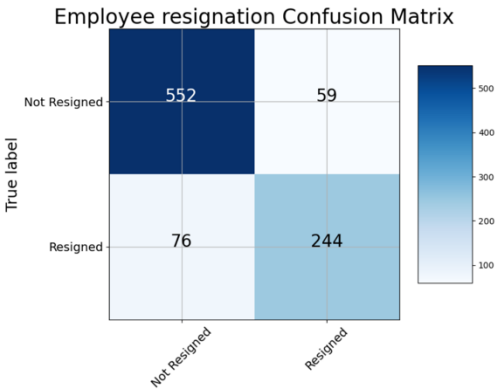
FINAL MODEL:

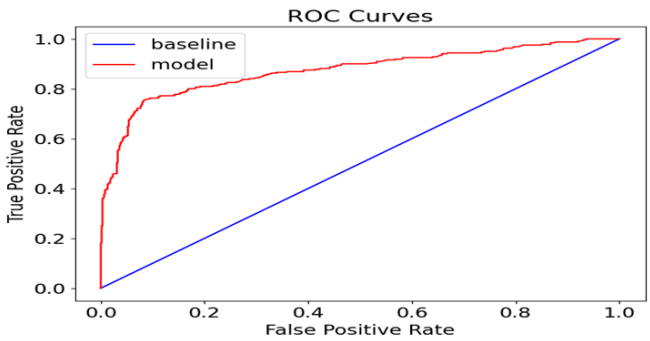
In our project, we emphasized the AUC score as our primary metric due to the class imbalance in our dataset. While the Random Forest model exhibited superior precision, we made a deliberate choice in favor of the SVM Classifier, utilizing all 12 features. This decision was rooted in our specific objective: identifying employees at risk of leaving rather than those likely to stay. SVM's strength in handling imbalanced data and its aptness for binary classification made it the optimal selection. This allowed us to concentrate on effectively pinpointing potential attrition risks.

Our final model achieved the following metrics:

AUC: 0.832
Precision: 0.805
Recall: 0.7625
Accuracy: 0.8549
F1 Score: 0.783

	Model Name	no of features	Features	AUC	Precision	Recall	Accuracy	F1 score
0	Logistic Regression	8	['JoiningYear', 'PaymentTier', 'Age', 'Education_Masters', 'City_New Delhi', 'City_Pune', 'Gender_Male', 'EverBenched_Yes']	0.664814	0.684729	0.434375	0.736842	0.531549
1	Decision Tree	11	['JoiningYear', 'PaymentTier', 'Age', 'ExperienceInCurrentDomain', 'Education_Bachelors', 'Education_Masters', 'Education_PHD', 'City_Bangalore', 'City_Pune', 'Gender_Male', 'EverBenched_Yes']	0.818011	0.845865	0.703125	0.853921	0.767918
2	SVM Classifier	12	['JoiningYear', 'PaymentTier', 'Age', 'ExperienceInCurrentDomain', 'Education_Bachelors', 'Education_Masters', 'Education_PHD', 'City_Bangalore', 'City_New Delhi', 'City_Pune', 'Gender_Male', 'EverBenched_Yes']	0.832968	0.805281	0.762500	0.854995	0.783307
3	SGD Classifier	12	['JoiningYear', 'PaymentTier', 'Age', 'ExperienceInCurrentDomain', 'Education_Bachelors', 'Education_Masters', 'Education_PHD', 'City_Bangalore', 'City_New Delhi', 'City_Pune', 'Gender_Male', 'EverBenched_Yes']	0.644801	0.633333	0.415625	0.716434	0.501887
4	Random Forest Classifier	12	['JoiningYear', 'PaymentTier', 'Age', 'ExperienceInCurrentDomain', 'Education_Bachelors', 'Education_Masters', 'Education_PHD', 'City_Bangalore', 'City_New Delhi', 'City_Pune', 'Gender_Male', 'EverBenched_Yes']	0.822995	0.871595	0.700000	0.861439	0.776430





LIMITATIONS:

- **Limited Hyperparameter Optimization:** The project did not tune all the hyperparameters for all models, so there may be room for improvement in performance.
- **Data Imbalance:** Getting rid of class imbalance is hard, and even when people try, it can make it harder to predict the minority class.
- **Validation Dataset Use:** A validation dataset was used to test the model, but its size and how well it represents the whole population can affect how well the model works in real life.
- **Assumptions and Simplifications:** During preprocessing or modelling, the project may have made assumptions or simplifications that could have changed the results.
- **Interpretability:** Machine learning models that are chosen for their ability to predict may not be easy to understand, which makes it hard to explain what the models say.

REFERENCES:

(1) Yujian Liu, Yazhe Li & Dejun Xie (2023) *Implications of imbalanced datasets for empirical ROC-AUC estimation in binary classification tasks, Journal of Statistical Computation and Simulation,* DOI: 10.1080/00949655.2023.2238235

(2) "Mishra P, Singh U, Pandey CM, Mishra P, Pandey G. *Application of student's t-test, analysis of variance, and covariance." ("Short-term effects of occlusion therapy and optical correction on ...")* Ann Card Anaesth. 2019 Oct-Dec;22(4):407-411. doi: 10.4103/aca.ACA_94_19. PMID: 31621677; PMCID: PMC6813708.

(3) Belete, Daniel & D H, Manjaiah. (2021). *Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results.* International Journal of Computers and Applications. 44. 1-12. 10.1080/1206212X.2021.1974663.

(4) "Linardatos P, Papastefanopoulos V, Kotsiantis S. *Explainable AI: A Review of Machine Learning Interpretability Methods." ("A comparison of explainable artificial intelligence methods in the ...") Entropy (Basel).* 2020 Dec 25;23(1):18. doi: 10.3390/e23010018. PMID: 33375658; PMCID: PMC7824368.

(5) Jun, Z. (2021). *The Development and Application of Support Vector Machine.* Journal of Physics: Conference Series, 1748(5), 052006. https://doi.org/10.1088/1742-6596/1748/5/052006

(6) *Logistic Regression: Understanding Its Significance* - Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/.