IIIT-Bangalore

Machine Learning

AIM - 511

Professors:
PROF. Viswanath G
PROF. Sushree Behera

Submitted by:
Madhav Girdhar (IMT2022009)
Siddhesh Deshpande (IMT2022080)
Krish Patel (IMT2022097)

Machine Learning Project Part-1

1 Introduction

The primary goal is to analyze the employee exit status data and use models to predict or understand patterns that influence employee turnover. This analysis includes data preprocessing, exploratory data analysis (EDA), model building, and optimization through hyperparameter tuning.
Note: We will cover Logistic Regression in Part-2 of Project.

2 Data Processing Steps

- Handling Duplicate Values : We first analyze the dataset and drop duplicates value if present.
- Handling Null values :
    1. Calculated Null value percentage of each and every columns.
    2. Drop the corresponding row of particular columns who have their null value percentage between 0 to 5.
    3. Now for the numerical columns we replace their null values either by mean or mode by analyzing their KDE plots.
    4. And for categorical columns we use KNN imputer with nearest neighbor value = 5.
- Encoding Categorical Columns: we use label encoder from sklearn library to encode the categorical columns.
- Data Cleaning / Drop Irrelevant Columns : We first find correlation of each and every column with the response variable and then check the absolute value of correlation and drop the columns that have their correlation value less than 10^{-2}.
- Outliers : Here in this classification dataset, we have not removed outliers because :
    1. Robust Algorithms: Many classification algorithms, particularly tree-based models (e.g., Random Forest, XGBoost), are robust to outliers and focus on majority patterns, so outliers don't skew results significantly.
    2. Sometime Outliers helps to Preserve Minority Classes, or to create Effective class boundaries.

3 Models Used

- Decsion Tree Classifier
- Random Forest Classifier
- XG-Boost Classifier
- Ada-Boost Classifier
- Gradient Boosting Classifier
- K Nearest Neighbours Classifier

4 Hyperparameter Tuning

1. **Decsion Tree Classifier** :
   - Training accuracy (f1_score) : 0.73
   - Testing accuracy (kaggle) : 0.68
   - Best Parameter :
     (a) class_weight - 'balanced'
     (b) criterion - 'gini'
     (c) max_depth - 10
     (d) max_features - 'sqrt'
     (e) max_leaf_nodes - None
     (f) minsamples_leaf - 5
     (g) min_sample_split - 2
2. **Random Forest Classifier** :
   - Training accuracy (f1_score) : 0.78
   - Testing accuracy (kaggle) : 0.7326
   - Best Parameter :
     (a) max_depth - 10
     (b) max_features - 'sqrt'
     (c) min_sample_leaf - 4
     (d) min_sample_split - 2
     (e) n_estimators - 200
3. **XG-Boost Classifier** :
   - Training accuracy (f1_score) : 0.771
   - Testing accuracy (kaggle) : **0.74494**
   - Best Parameter :

(a) colsample_bytree - 1
(b) gamma - 0
(c) learning_rate - 0.1
(d) max_depth - 3
(e) min_child_weight - 1
(f) n_estimators - 200
(g) reg_alpha - 0.1
(h) reg_lambda - 2
(i) subsample - 0.8

4. **Ada-Boost Classifier** :

- Training accuracy (f1_score): 0.769
- Testing accuracy (kaggle) : 0.7427
- Best Parameter :
  (a) base_estimator_max_depth - 1
  (b) base_estimator_min_sample_split - 2
  (c) learning_rate - 1
  (d) n_estimator - 50

5. **Gradient Boosting Classifier** :

- Training accuracy (f1_score) : 0.777
- Testing accuracy (kaggle) : 0.739
- Best Parameter :
  (a) learning_rate - 0.1
  (b) max_depth - 5
  (c) max_features - 'sqrt'
  (d) min_sample_split - 2
  (e) n_estimators - 100
  (f) subsample - 1

6. **K Nearest Neighbours Classifier** :

- Training accuracy (f1_score): 0.676
- Testing accuracy (kaggle) : 0.509
- Best Parameter :
  (a) metric - 'manhattan'
  (b) n_neighbors - 11
  (c) weight - 'uniform'

5 Discussion on the performance of different approaches

- **Decision Tree Classifier:** Offers decent interpretability but struggles with generalization. Although it reached moderate training accuracy (0.73), its lower testing accuracy (0.68) indicates overfitting, which is common for single decision trees in complex datasets.
- **Random Forest Classifier:** Combines multiple decision trees to reduce overfitting, achieving better stability and generalization with a testing accuracy of 0.7326. Random Forest's ensemble approach helps it capture more complex patterns without sacrificing accuracy on unseen data, making it robust for this task.
- **XG-Boost Classifier:** Achieved the highest testing accuracy (0.74494), showcasing its ability to handle complex patterns and interactions in the data. XGBoost's regularization parameters help prevent overfitting, while its gradient-boosting method optimizes each tree iteration for high accuracy, making it an excellent choice for predictive modeling in structured datasets.
- **Ada-Boost Classifier:** While effective for improving accuracy on imbalanced data by focusing on misclassified samples, it showed slightly lower performance than XGBoost (testing accuracy: 0.7427). AdaBoost can struggle with noisy data or outliers, as it places more weight on difficult-to-classify samples, which can impact overall stability.
- **Gradient Boosting Classifier:** Similar in approach to XGBoost but slightly less optimized, resulting in lower accuracy (0.739) compared to XGBoost. Gradient Boosting works well on complex datasets but can be slower and more prone to overfitting if not carefully tuned, especially when using deeper trees.
- **K-Nearest Neighbors (KNN) Classifier:** Has the lowest testing accuracy (0.509), indicating it is less suitable for this dataset. KNN tends to perform poorly in high-dimensional or large datasets due to its sensitivity to feature scaling and inability to capture complex boundaries effectively.

6 Experiments

In our experiments, we mainly focused on XGBoost and AdaBoost, as they demonstrated the best predictive accuracy for employee exit status. XGBoost achieved the highest testing accuracy of 0.74494, leveraging its gradient-boosting approach and regularization to capture complex patterns in the data effectively. AdaBoost followed closely with an accuracy of 0.7427, benefitting from its focus on harder-to-classify instances.

1. **Experiment-1 :** During preprocessing, we retained all columns without dropping any features and applied standard scaling to normalize the data. This approach yielded an accuracy of **0.74583** on the Kaggle test dataset, demonstrating the effectiveness of preserving all features in conjunction with standardized scaling.

2. **Experiment-2 :** In this experiment,we retained all columns without dropping any features and a new column, leave, was created to analyze employee exit tendencies based on the features distance from home and remote work. The hypothesis was that employees living significantly farther from the workplace and not offered remote work would have a higher likelihood of leaving the company. To model this, a threshold of 40 km was set for distance from home. An XGBoost classifier was utilized, with hyperparameter tuning applied as previously described except that n_estimator which we put 400 . This approach yielded a test accuracy of **0.74724**.

3. **Experiment-3 :**Subsequently, the analysis was extended by employing an AdaBoost classifier to evaluate employee exit tendencies with a refined approach. In this iteration, the threshold for distance from home was adjusted to 35 km, under the assumption that a slightly lower threshold might better capture the relationship between commuting distance and the likelihood of leaving the company. The AdaBoost classifier was chosen for its ability to enhance model performance through iterative boosting. After performing hyperparameter tuning, the optimal value for n_estimators was identified as 100. This updated approach achieved a slightly improved test accuracy of **0.74767**, indicating the effectiveness of the adjustments in capturing patterns within the data.

4. **Experiment-4 :**Building upon the same features and insights, the XGBoost classifier was employed once again to analyze employee exit tendencies. This model iteration retained the threshold for distance from home at 35 km, aligning with the refined approach used in the AdaBoost classifier. XGBoost was selected for its robust handling of structured data and superior performance in predictive modeling tasks. After hyperparameter tuning, the n_estimators was set to 400, ensuring the model's effectiveness in capturing complex patterns in the data. This approach resulted in a test accuracy of **0.74848**, demonstrating its competitive performance and validating the relevance of the selected features and refined threshold.

7 Code

All the codes for preprocessing and model fitting and prediction and experiments done are here Click here

Machine Learning Project Part-2

1. Introduction

In this phase of our Machine Learning project, we aim to implement and evaluate three powerful predictive models: Logistic Regression, Support Vector Machines (SVM), and Neural Networks. Logistic Regression serves as a foundational linear model, well-suited for binary classification tasks and providing interpretable results. SVM offers a robust approach to classification by identifying optimal hyperplanes, especially effective for datasets with clear margins of separation. Neural Networks, on the other hand, bring the ability to model complex, non-linear relationships through their multi-layer architecture.

2. Data Processing Steps

1. **Handling Outliers:**Outliers in the dataset were addressed by adjusting their values to fall within the interquartile range (IQR) for each column where outliers were identified. This ensures the data remains representative and minimizes the influence of extreme values on model performance.

2. **Imputation of Missing Values:**Imputation of Missing Values: Null values were handled systematically to maintain data integrity. For numerical columns, missing values were filled using the mean, while for categorical columns, the mode was used. This approach preserves the overall distribution and avoids introducing bias into the dataset.

3. **Feature Scaling and Encoding:**To prepare categorical variables for machine learning models, One-Hot Encoding was applied. This method effectively transformed categorical values into numerical form while retaining the original information, enabling better compatibility with the models. And along with that MinMax scale is used to scale the data for better result.

3. Model Used

- Logistic Regression
- Support Vector Machine
- Neural Network

4. Model Performance

1. **Logistic Regression:**

- Training Accuracy : 0.759
- Test Accuracy (Kaggle) : 0.74726
- Best Parameter :
  (a) Regularization technique : 'l2'
  (b) Solver : 'lbfgs'
  (c) Regularization parameter : 1
  (d) l1_ratio : 0.1
  (e) Max Iteration : 500

2. **Support Vector Machine:**

   - Training Accuracy : 0.759
   - Test Accuracy (Kaggle) : 0.74790
   - Best Parameter :
     (a) Kernel : 'linear'
     (b) gamma : 'scale'
     (c) Regularization parameter : 1

3. **Neural Network:**

   - Training Accuracy : 0.7461
   - Test Accuracy (Kaggle) : 0.73910
   - Best Parameter :
     (a) 1st layer with 64 Neurons, Activation : 'relu'
     (b) 2nd layer with 32 Neurons, Activation : 'relu'
     (c) Output layer with 2 Neurons, Activation : 'softmax'
     (d) Optimizer : 'adam'
     (e) Loss : 'sparse_categorical_crossentropy'

5. Discussion on Performance of different Approaches

1. **Logistic Regression :**It achieved a training accuracy of 0.759 and a test accuracy of 0.74726. With the chosen regularization technique (l2), solver (lbfgs), and a regularization parameter of 1, the model performed well on both the training and test datasets. This indicates that the model generalizes effectively without significant overfitting. Logistic Regression's simplicity and interpretability make it a strong baseline model for classification tasks, but its linear nature might limit its performance on datasets with complex, non-linear relationships.

2. **Support Vector Machine :** The SVM model achieved a training accuracy of 0.759 and a test accuracy of 0.74790, slightly outperforming Logistic Regression. During experimentation, various kernels, including RBF and polynomial, were implemented; however, the linear kernel provided the best performance. This suggests that the data is likely linearly separable or that the linear kernel offered a better trade-off between accuracy and computational efficiency. With gamma set to 'scale' and a regularization parameter of 1, the model successfully identified an optimal hyperplane for classification, ensuring robust performance on the test data.

3. **Neural Network :** The NN model achieved a training accuracy of 0.7461 and a test accuracy of 0.73910, which are slightly lower than both Logistic Regression and SVM. After extensive experimentation with various architectures, the best performance was obtained using two hidden layers with 64 and 32 neurons, respectively, each using ReLU activation, and a softmax output layer. The model was optimized with the Adam optimizer and sparse categorical cross-entropy loss. This configuration was determined to provide the optimal balance between model complexity and performance, effectively capturing patterns in the data while avoiding overfitting.