Predicting Employee Attrition

BY - ANSHIKA CHAUHAN

Attrition Analytics – Employee Attrition Prediction Report

Introduction

Employees attrition, the loss of employees through natural turnover, resignation, or retirement, is a significant concern for organizations. High attrition rates can lead to increased recruitment and training costs, loss of productivity, and negative impacts on morale and team dynamics. Therefore, developing predictive models to identify factors contributing to attrition and accurately forecast employee turnover is crucial for effective human resource management.

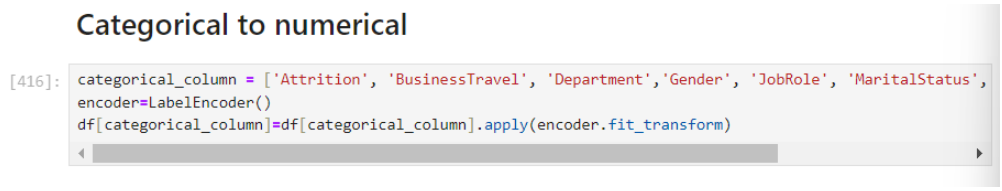
Dataset Analysis

The dataset used in this analysis is the "WA_Fn-UseC_-HR-Employee-Attrition" dataset, which contains various features related to employees in a company, including demographics, job roles, satisfaction levels, and attrition status.
Upon loading the dataset and performing initial analysis, it was observed that there were no missing values in the dataset. However, categorical variables needed to be encoded numerically for modeling purposes.

Preprocessing Steps

To prepare the dataset for modeling, several preprocessing steps were undertaken:

a. Categorical to numerical conversion: Categorical variables such as 'BusinessTravel', 'Department', 'Gender', etc., were encoded using label encoding to convert them into numerical format.
b. Feature selection: Unnecessary columns such as 'EmployeeCount', 'EmployeeNumber', 'Over18', and 'StandardHours' were dropped from the dataset. Additionally, oversampling was applied using the RandomOverSampler to handle class imbalance.
c. Splitting dataset: The dataset was split into training and testing sets using a 70-30 split ratio.



Split dataset into train and test data

[420]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Model Development

A Random Forest classifier was chosen as the predictive model due to its ability to handle complex relationships in data, handle both numerical and categorical features, and provide feature importances.

The initial Random Forest model was trained on the training data, and its performance was evaluated using metrics such as accuracy, precision, recall, and confusion matrix.

Applying Random Forest

[422]:
```
def train_predict_evaluate(model,X_train,y_train,X_test):
  model.fit(X_train,y_train)
  y_pred = model.predict(X_test)
  print("Accuracy: ",accuracy_score(y_test,y_pred))
  print("Precision: ",precision_score(y_test,y_pred))
  print("Recall: ",recall_score(y_test,y_pred))
  print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
```

[423]:
```
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest Classifier
# Setting random state for randomization
rf = RandomForestClassifier(random_state=42)
```

Evaluation Results

The initial Random Forest model achieved satisfactory performance with reasonable accuracy, precision, and recall scores. However, hyperparameter tuning was performed using both RandomizedSearchCV and GridSearchCV to optimize the model further.

[424]:
```
from sklearn.model_selection import RandomizedSearchCV

# Taking a huge range of values to get nearer to optimal parameters

# Parameters to check
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 150, num = 10)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

# Creating random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

[432]:
```
# Using the random grid to search for best hyperparameters
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                               n_iter = 100, cv = 3, verbose=2,
                               random_state=42, n_jobs = -1)
```

[433]:
```
# Training model and printing results
train_predict_evaluate (rf_random, X_train, y_train, X_test)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
Accuracy:  0.9783783783783784
Precision:  0.9796954314720813
Recall:  0.9796954314720813
Confusion Matrix:
 [[338   8]
 [  8 386]]
```

[434]:
```
# Finding the best parameters
rf_random.best_params_
```

[434]:
```
{'n_estimators': 400,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 103,
 'bootstrap': False}
```

Optimization Techniques Used

a. RandomizedSearchCV: A randomized search was conducted over a predefined hyperparameter space to find the optimal combination of hyperparameters. This approach helps in exploring a wide range of hyperparameters efficiently.
b. GridSearchCV: Following the RandomizedSearchCV, a grid search was performed to further refine the hyperparameters based on the best parameters found from the randomized search.

Insights Gained

The optimized Random Forest model demonstrated improved performance compared to the initial model, indicating the importance of hyperparameter tuning in enhancing model accuracy and effectiveness.

Challenges Encountered

One of the main challenges encountered during the analysis was dealing with class imbalance in the dataset. This was addressed using oversampling techniques to ensure that the model learns equally from both classes.

Recommendations for Reducing Employee Attrition

Based on the analysis and insights gained, the following recommendations are proposed for reducing employee attrition:

a. Conduct regular employee satisfaction surveys to identify factors contributing to dissatisfaction and address them proactively.
b. Provide opportunities for career advancement, skill development, and training to enhance employee engagement and retention.
c. Implement flexible work arrangements and work-life balance initiatives to improve employee well-being and job satisfaction.
d. Foster a positive work culture, open communication, and recognition programs to boost morale and motivation among employees.
e. Continuously monitor employee turnover trends and analyze exit interviews to identify recurring patterns and take corrective actions accordingly.

Conclusion

In summary, predicting when employees might leave is very important for companies to handle their staff well. By using methods like machine learning, companies can figure out why people might leave and come up with smart plans to stop them from going.

This report explains all the steps we took to study the data, get it ready for the computer, make a clever model, test how good it is, and make it even better. We also suggest some ideas for keeping employees happy and sticking around.

End of Report