DAT 430 Project 2: Part 1

Daniel Palhano

Southern New Hampshire University

DAT 430

Venkata Kotra

October 19, 2025

Engineered Features

The features we have engineered for the project have revolved all around looking into attrition and how we can trim specific categorical data types to allow for easy analysis for the model. The first two features we created for this project that will then be placed in our respective baseline and predictive models are the 'high_income' and 'Tenured' features as seen in figure 1. These features will allow us to be able to put filters on what will classify an employee as a high income earner if they earn 7000 or more a month and tenured if they have worked 11 or more years in the company. The next thing done after creating these features is copying the data set with the features we want to focus on for our analysis. These features we selected are Age, Attrition, Department, Education, Gender, high_income, JobRole, JobSatisfaction, MaritalStatus, MonthlyIncome, NumCompaniesWorked, PerformanceRating, StockOptionLevel, tenured, TotalWorkingYears, TrainingTimesLastYear, YearsAtCompany, YearsInCurrentRole, and YearsWithCurrManager. The reason for choosing these features are mainly due to the position of the question from the leadership team which is lack of engagement. This lack of engagement could come from many of these features we copied when engineering our 2 features.

Figure 1

Creation of the 'high_income' and 'Tenured' features

```
# 2-1 Create High Income feature
hr_merged_df['high_income'] = 0
hr_merged_df.loc[hr_merged_df['MonthlyIncome'] >= 7000, 'high_income'] = 1

# 2-2 Create Tenured feature
hr_merged_df['Tenured'] = 0
hr_merged_df.loc[hr_merged_df['YearsAtCompany'] >= 11, 'Tenured'] = 1

# 2-3 Copy Data Set with a Subset of Features
data_model_df = hr_merged_df[['Age', 'Attrition', 'Department', 'Education', 'Gender',
                              'high_income', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
                              'MonthlyIncome', 'NumCompaniesWorked', 'PerformanceRating',
                              'StockOptionLevel', 'Tenured', 'TotalWorkingYears', 'TrainingTimesLastYear',
                              'YearsAtCompany', 'YearsInCurrentRole', 'YearsWithCurrManager']].copy()
data_model_shape = data_model_df.shape
data_rows, data_cols = data_model_shape[0], data_model_shape[1]
print(f'Feature Engineering Data: rows = {data_rows}, cols = {data_cols}')
```

```
Feature Engineering Data: rows = 1470, cols = 19
```

Another feature that was created during the project was using the ‘One Hot Encoder’ which changed Gender, JobRole, Department, and MaritalStatus all into binary types as seen in figure 2. Doing this simply changed the data from the categorical numeric values from the index it was given to a binary form which turns the index to 0 and 1 instead (Saxena, 2020).

Figure 2

One Hot Encoder transforming features from categorical to binary

```
def onehot_encoder(df, column_names):
    """
    Args:
        df (pd.DataFrame): The input DataFrame.
        columns_names (list): A list of column names to encode.

    Returns:
        pd.DataFrame: The DataFrame with specified columns encoded.
    """
    encoder = OneHotEncoder(sparse=False)
    one_hot_encoded = encoder.fit_transform(df[column_names])
    one_hot_df = pd.DataFrame(one_hot_encoded, columns=encoder.get_feature_names_out(column_names))
    df_encoded = pd.concat([df, one_hot_df], axis = 1)
    df_encoded = df_encoded.drop(column_names, axis = 1)

    return df_encoded

# 3-2 Run the function on Categorical variables in the data set
data_df = onehot_encoder(data_model_df, ['Gender', 'JobRole', 'Department', 'MaritalStatus'])
data_shape = data_df.shape
data_rows, data_cols = data_shape[0], data_shape[1]
print(f'Encoded Data: rows = {data_rows}, cols = {data_cols}')
```

```
Encoded Data: rows = 1470, cols = 32
```

Establishing a Baseline

When establishing a baseline for our project, we need to pick a model that will allow us to utilize many of our variables that we have selected while still being able to predict more complex relationships and correlations between our features (US EPA, 2024). The use of the random forest will allow us to model in a non-linear model showing these complex relationships over the linear modeling of logistic regression. Looking down at figure 3, there is the random forest model that was chosen to be the baseline for our predictions.

Figure 3

Random Forest Model with Metrics

```
model = RandomForestClassifier(random_state = 67)

## Hyperparameter Tuning & Fit
param_grid = {  "n_estimators"      : [250, 300],
                "max_depth"         : [5, 10],
                "bootstrap": [True, False],
                "class_weight": ['balanced']}
rf_grid_search = GridSearchCV(model, param_grid, n_jobs = -1, cv = 2)
rf_grid_search.fit(X_train, y_train)


# 5-2 Predict on the Train and Test Sets
y_rf_train_pred = rf_grid_search.predict(X_train)
y_rf_test_pred = rf_grid_search.predict(X_test)


# 5-3 Report Metrics
# accuracy_score, precision_score, recall_score
train_accuracy = accuracy_score(y_train, y_rf_train_pred)
train_precision = precision_score(y_train, y_rf_train_pred)
train_recall = recall_score(y_train, y_rf_train_pred)
print(f"\ntrain set metrics")
print(f"accuracy = {round(train_accuracy, 2)}")
print(f"precision = {round(train_precision, 2)}")
print(f"recall = {round(train_recall, 2)}")

test_accuracy = accuracy_score(y_test, y_rf_test_pred)
test_precision = precision_score(y_test, y_rf_test_pred)
test_recall = recall_score(y_test, y_rf_test_pred)
print(f"\ntest set metrics")
print(f"accuracy = {round(test_accuracy, 2)}")
print(f"precision = {round(test_precision, 2)}")
print(f"recall = {round(test_recall, 2)}")
```

```
train set metrics
accuracy = 0.97
precision = 0.99
recall = 0.92

test set metrics
accuracy = 0.86
precision = 0.82
recall = 0.56
```

The original business prediction on employee attrition was the concerns that the cause came from lack of employee engagement and overall job satisfaction. Looking at the new metrics we see with the selected features that were chosen for the new model that with our test metrics we have an accuracy of 0.86, precision of 0.82, and recall of 0.56. This tells us the overall recall of 0.56 being relatively good with a precision pretty high as well. We have a solid model which does much better compared to the previous logistic models we used in the previous labs utilizing the HR datasets while being able to use the many features we needed in that we copied earlier in a subset of features when we engineered tenured and high_income.

Predictive Modeling

Moving on from our random forest model we chose to use for our baseline, the second predictive model we decided to use was ada boost classification. The reason for choosing ada boost over other models is the versatility when it comes to weighting itself. Over the many

estimators it goes through, the weighting changes over time to give us back a more accurate prediction overall (GeeksforGeeks, 2019).

Figure 4

Ada Boost Classification Model with Metrics

```
model = AdaBoostClassifier(random_state = 67)

## Hyperparameter Tuning & Fit
param_grid = {  "n_estimators"   : [250, 300],
                "learning_rate"  : [0.1, 0.01, 0.001]}

ada_grid_search = GridSearchCV(model, param_grid, n_jobs=-1, cv=2)
ada_grid_search.fit(X_train, y_train)

# 6-2 Predict on the Train and Test Sets
y_ada_train_pred = ada_grid_search.predict(X_train)
y_ada_test_pred = ada_grid_search.predict(X_test)


# 6-3 Report Metrics
# accuracy_score, precision_score, recall_score
train_accuracy = accuracy_score(y_train, y_ada_train_pred)
train_precision = precision_score(y_train, y_ada_train_pred)
train_recall = recall_score(y_train, y_ada_train_pred)
print(f"\ntrain set metrics")
print(f"accuracy = {round(train_accuracy, 2)}")
print(f"precision = {round(train_precision, 2)}")
print(f"recall = {round(train_recall, 2)}")

test_accuracy = accuracy_score(y_test, y_ada_test_pred)
test_precision = precision_score(y_test, y_ada_test_pred)
test_recall = recall_score(y_test, y_ada_test_pred)
print(f"\ntest set metrics")
print(f"accuracy = {round(test_accuracy, 2)}")
print(f"precision = {round(test_precision, 2)}")
print(f"recall = {round(test_recall, 2)}")
```

```
train set metrics
accuracy = 0.86
precision = 0.92
recall = 0.56

test set metrics
accuracy = 0.85
precision = 0.82
recall = 0.5
```

Between the ada boost and random forest models, we see a combination of predictive modeling for our selected features and those we engineered. The business's goal is to figure out whether or not job satisfaction plays a role within attrition and engagement as well. Engagement for our analysis can be anything from their role within the company from such as employee performance ratings, number of years an employee has been with the company, their current role, even stock option level, and even years with their current manager. These features all play a factor in the engagement of employees that were not directly considered by the leadership team in what could be impacting attrition for the company.

We see the variation in the metrics of our predictive models we chose. Starting with the random forest we were able to get a training set accuracy of 0.97, precision of 0.99, and recall of 0.92; for our testing set we have an accuracy of 0.86, precision of 0.82, and recall of 0.56. For our ada boost model we got back for our training set accuracy of 0.86, precision of 0.92, and

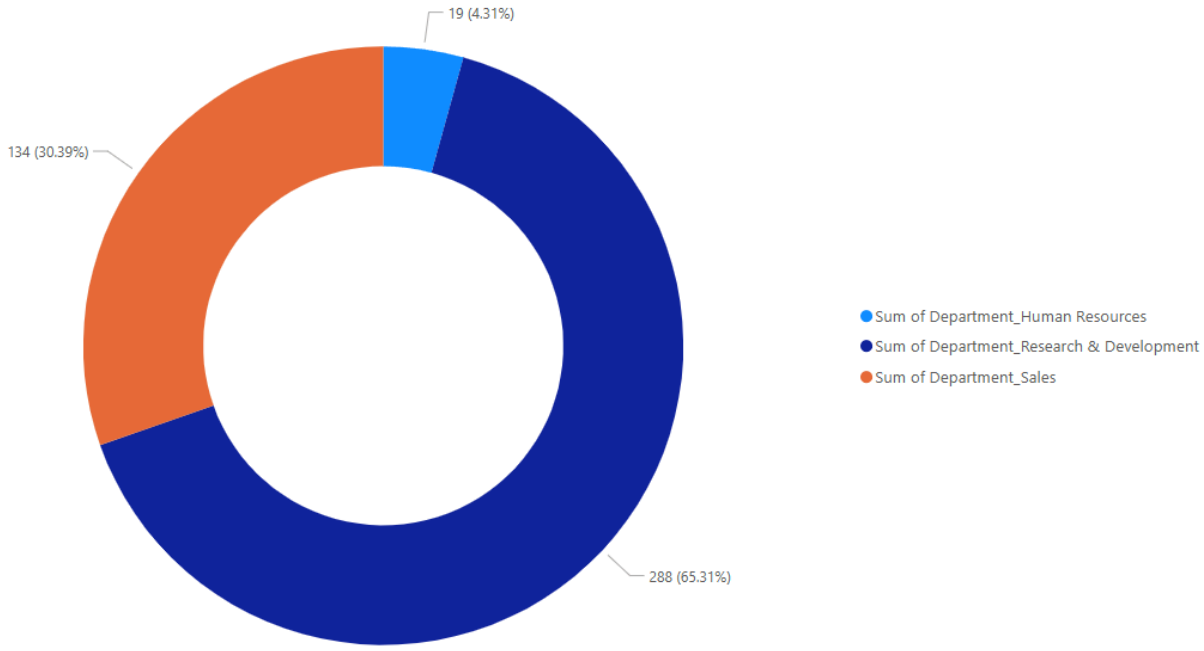
recall of 0.56; for our testing set we have an accuracy of 0.85, precision of 0.82, and recall of 0.50. The key takeaways for our data mainly revolve around the random forest mode which shows our training set being a little too good across the three metrics. This means the model could be overfitting due to the high recall and precision as well (GeeksforGeeks, 2023). This is seen to change with the testing set which then the recall drops to nearly half of what it was in the training set and more than 10 points for both accuracy and precision. The ada boost model is slightly different and shows more tame numbers especially on recall, meaning it is more cautious about its prediction but still is precise. Overall, the main takeaway is the close numbers on the testing sets, but the random forest could be suffering from overfitting as seen in the training set metrics we got back.

Can We Accurately Predict Outcomes?

The original question from the leadership is what factors play into attrition for the company. We can predict attrition within the company with relative accuracy, but the issue is what employees are not being accounted for when they do leave. For example, our ada boost model can predict attrition precisely, same with the random forest, but the recall of the ada boost is still ‘low’ meaning it still does not catch all instances of leaving. We can use the data we gained to make visuals which could point to attrition and how specific factors do affect it and what types of signs the leadership team can look at which point to one making their exit.

Figure 5

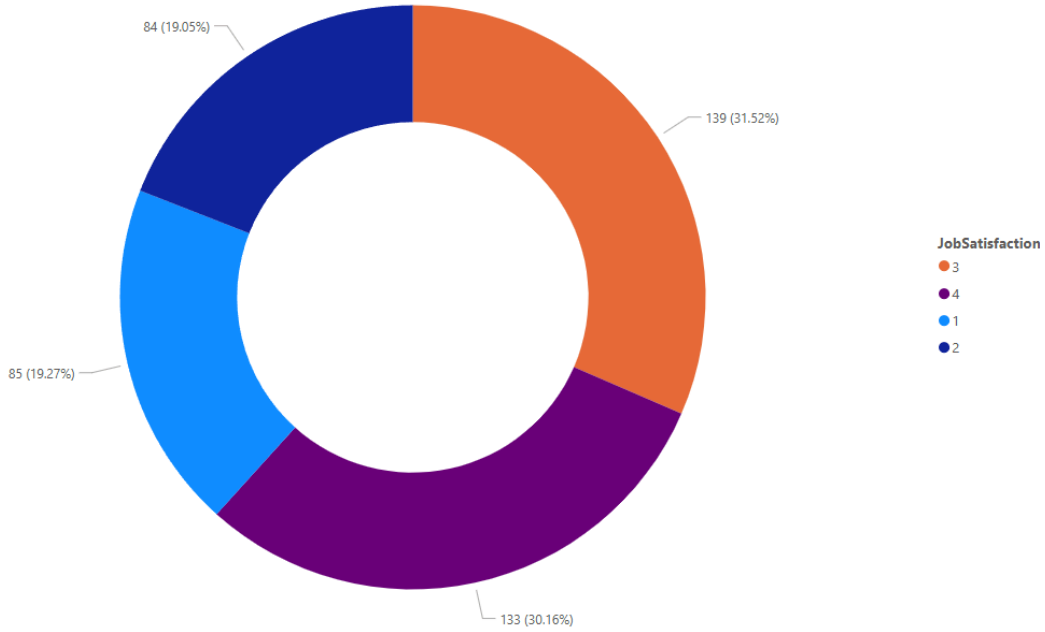
Donut chart of employee breakdown per department



Looking up towards figure 5, we see a breakdown of the number of employees we have for our final prediction data set. Now the reason we must look at our analysis in terms of departments rather than all together is due to the differing roles, responsibilities for these roles, and even management within the departments. Sales and R&D departments both are the backbone of the company which both are based on performance and their performance affects the company, while HR deals with internal affairs which keeps peace and balance within the company. Keeping the differences of the departments in mind will allow us to separate them individually to look at the defining factors that could cause employees to leave. One feature we can look at regarding all employees regardless of department is the job satisfaction. Looking down towards figure 6 we see the breakdown of job satisfaction across the 3 departments we are looking at showing the majority of employees, nearly 62%, rating their satisfaction at least a 3 or higher. This on the other hand shows a concerning 38% not favoring their job highly, either a 2 or lower which could be a point of concern for the company.

Figure 6

Donut chart for job satisfaction for all employees



References

GeeksforGeeks. (2024, February 22). *Random Forest Algorithm in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning

GeeksforGeeks. (2023, November 29). *Training data vs Testing data*. GeeksforGeeks. https://www.geeksforgeeks.org/python/training-data-vs-testing-data

Wexler, S., Shaffer, J., & Cotgreave, A. (2017). *The Big Book of Dashboards : Visualizing Your Data Using Real-World Business Scenarios*. John Wiley & Sons.

US EPA. (2024, June 6). *Random Forest Modeling | US EPA*. United States Environmental Protection Agency. https://www.epa.gov/streamflow-duration-assessment/random-forest-modeling

GeeksforGeeks. (2019, May 3). *AdaBoost in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/AdaBoost-in-Machine-Learning

Saxena, S. (2020, August 13). *What are Categorical Data Encoding Methods | Binary Encoding*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/#h-one-hot-encoding