DSC 630-T301

Prediction of High Performing Employees' Job Satisfaction

Julie Campbell

Bellevue University

Predictive Analytics 630

Andrew Hua

Introduction

Companies implement employee engagement surveys at a frequency of their choosing (e.g., monthly, quarterly, yearly) to predict the behavior of their employees. In particular, the job satisfaction scores can tell the employer how fulfilled the employees feel with their day-to-day tasks and their work conditions, such as pay or work environment. High job satisfaction can lead to lower turnover rates, increased productivity, positive work culture, and therefore increased profits (Indeed, 2022). The employer must consider what actions they are willing to take to improve conditions to avoid low satisfaction scores, especially for their top performers.

A model will be created to predict when high performing employees may have low or medium job satisfaction. This information will be used to distribute allocated resources to improve their satisfaction with their position. The cost of replacing an existing employee can range from 1.5x to 2x of their annual salary (Bersin, 2013). Although turnover can be expected, the cost of losing a high performer can have the potential to be higher than average. This proposed model would have the most benefit if used by human resource departments so that they can adjust their strategies to retain employees as needed.

The dataset that will be used for model creation was procured on Kaggle and was artificially created originally by IBM data scientists to uncover the factors that lead to employee attrition (Sahoo, 2020). It contains features related to an employee's human resource profile such as department, job role, pay, and employee engagement survey scores. The job satisfaction score will be the target of the model. It is rated on a scale of 1-4 where 1 is low and 4 is very high. This type of data would be available to most companies and industries from their human resources department.

Methods/Results

Exploration

The original dataset was uploaded as train and test files; however, both files were consolidated for random sampling. After consolidation, 1,470 employee records and 34 features were found. The describe function and histograms were used to confirm that no null values or outliers were present in the data set. A unique employee number was generated to anonymize the employee until further identification would need to be done for retention purposes. The employee count, over 18, and standard hours features were found to only consist of one value (ex: 1, Y, 80).

The number of records associated with job satisfaction was unbalanced with more records being related to high (3) or very high (4) satisfaction. The performance rating also shows an imbalance with many employees being rated as excellent (3) performers and no records below an excellent (3) rating.

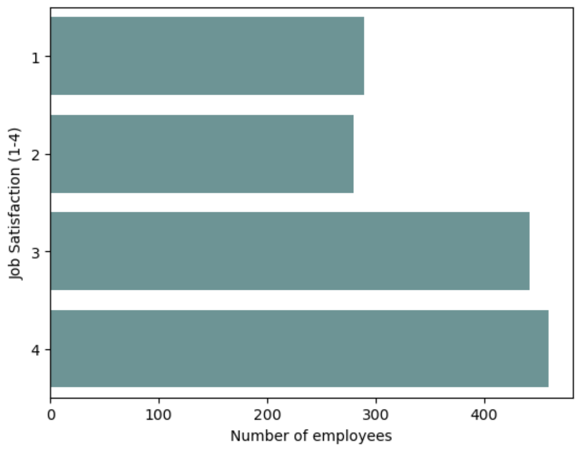


Figure 1 - Job Satisfaction Score Distribution

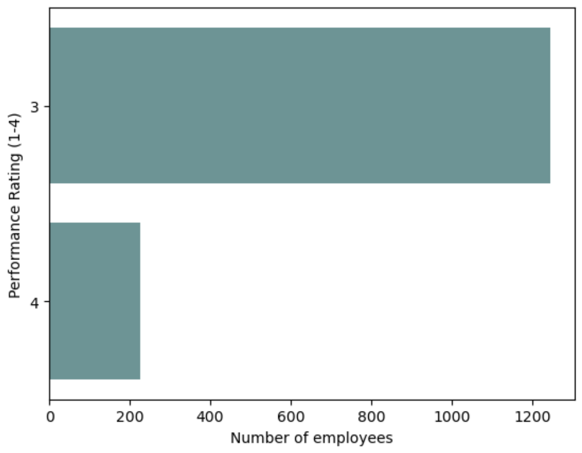


Figure 2 - Performance Rating Distribution

Multiple values related to pay were found including hourly rate, daily rate, monthly rate, and monthly income. A correlation matrix between these variables was reviewed, and the result showed that each variable had a low negative or positive correlation to each other.

	Daily Rate	**Hourly Rate**	**Monthly Rate**	**Monthly Income**
Daily Rate	1.0	0.02	-0.03	0.01
Hourly Rate	0.02	1.0	-0.02	-0.02
Monthly Rate	-0.03	-0.02	1.0	0.03
Monthly Income	0.01	-0.02	0.03	1.0

Table 1 - Pay Rate Correlation Matrix

In addition, a correlation heatmap was created for numeric values with more than one possible value.

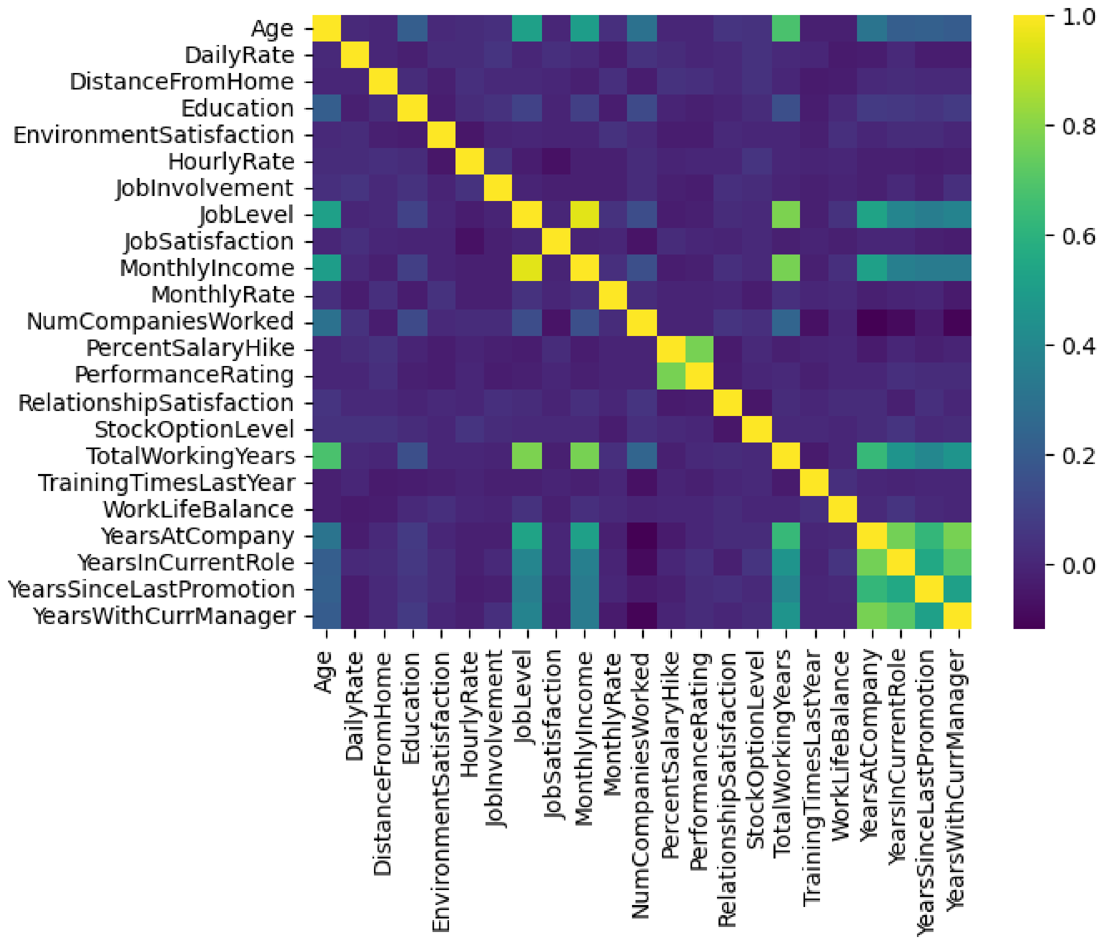


Figure 3 - Correlation Matrix

The following features were found to have a high correlation to one another: total working years, years with current manager, years in current role, years since last promotion, and years at company. Job level, age, and monthly income are also strongly correlated.

Preparation

The employee number, employee count, standard hours, and over 18 were removed as they either contained only one value or were not useful for model creation. Due to high correlation, the following features were also removed: age, years with current manager, years in role, years since last promotion, years at company, and monthly income. This decision was made based on a later evaluation of the features level of association with the target variable. Years at company and job level provided the most value to model creation from the highly correlated features.

The numeric and categorical features were split and then merged after encoding the categorical features (e.g. Business Travel, Department, Education Field, Gender, Job Role, Marital Status, and Overtime). To narrow down the most effective use of resources, job satisfaction was reclassified into three levels on a scale of 1-2.

Class	**Job Satisfaction**				**Performance**	
	1 - Low	2 - Medium	3 - High	4 - Very High	3 - Excellent	4-Outstanding
0			🟩	🟩	🟩	🟩
1	🟩					🟩
2	🟩	🟩			🟩	🟩

Table 2 - Target Classification

Specifically, our main target is the employees in Class 1 where job satisfaction is low (1) and performance is outstanding (4). If additional resources were available, we would then address Class 2 that includes the rest of the employees that scored low (1) to medium (2) job satisfaction.

The dataset was then split into train (60%) and test (40%) sets. The SMOTE methodology was used to balance the train set to randomly increase the minority class (class 1 and 2) examples by replicating them. The chi-square test of independence was used on the balanced train set to narrow down the features that had a significant association with job satisfaction. Features with p-values resulting from the test that had a value of <= .05 (significant) were selected for the model.

Features Selected	**p-value**
Daily Rate, Distance from Home, Environment Satisfaction, Job Level, Hourly Rate, Monthly Rate, Percent Salary Hike, Relationship Satisfaction, Stock Option Level, Total Working Years	0.0000
Education	0.0016
Job Involvement	0.0012
Overtime	0.0044
Gender	0.0239

Table 3 - Features Selected for Model and p-values

Model Selection and Evaluation Methods

The model types that were chosen for evaluation are decision tree, k neighbors, multinomial logistic regression, random forest model, and xgboost. They were chosen based on their ability to handle multi-class classification. The GridSearchCV function was used for each model type to get the best hyperparameters per model for evaluation. A pipeline was used with a min-max scaler and the associated classifier details would be injected. The min-max scaler was specifically used to normalize the continuous variables to reduce bias. Metrics that were chosen to evaluate each model were accuracy score, confusion matrix, precision, recall, f1-score, and ROC AUC. The accuracy score is a measure of how well a model can correctly classify all records. A confusion matrix and classification report (precision, recall, f1-score) will show the true positive, true negative, false positive, and false negatives that resulted from the model. The ROC AUC value will provide the probability that a positive instance would be ranked higher than a negative one.

As this is a multi-class model, the macro average was utilized from the classification report to ensure that the average doesn't prioritize satisfied employees (class 0).

Model	Accuracy	Macro Average			ROC AUC
		Precision	Recall	F1-Score	
Decision Tree Classifier	49%	40%	41%	40%	0.55
K Neighbors Classifier	49%	41%	44%	42%	0.58
Multinomial Logistic Regression	47%	42%	58%	44%	0.77
Random Forest Model	54%	45%	42%	42%	0.76
XGBoost Classifier	52%	45%	43%	43%	0.71

Table 4 - Model Metric Results

The results showed that the random forest model had the best accuracy and precision across all classes, and the multinomial logistic regression had the best recall, f1-score, and ROC AUC. The next step for model evaluation is to determine how it specifically predicts those that were assigned class 1 or 2.

Model	Class 1: Requires Immediate Action			Class 2: At-Risk		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Decision Tree Classifier	23%	26%	24%	38%	39%	39%
K Neighbors Classifier	24%	35%	28%	41%	44%	42%
Multinomial Logistic Regression	26%	83%	39%	41%	42%	41%
Random Forest Model	33%	26%	29%	43%	24%	31%
XGBoost Classifier	35%	30%	33%	41%	31%	35%

Table 5 – Class 1 and 2 Model Predictions

The XGBoost classifier showed the highest precision of predicting our main target (class 1); however, the recall and f1-score are low which could lead to resources going to employees that were already satisfied. The multinomial logistic regression model has relatively high recall and f1-scores for both class 1 and 2. The decision tree and k neighbors classifiers appear to have some of the lowest performances all around.

Conclusion

After reviewing the features that were selected due to a significant association with job satisfaction, the human resources depart should have the ability to propose that resources should be allocated to improve pay, work environment, relationship management, and provide additional stock options for high performers that have low to medium job satisfaction. The models evaluated had a significant tradeoff between precision and recall. Recall must take priority as resources should only be used for retaining high performing employees that may have low job satisfaction.

Based on the model choices evaluated, the multinomial logistic regression model has one of the highest recall values while still retaining precision within range of other models. Implementing this model would be better than no action being taken on the high performers with low job satisfaction. Replacing a high performer would financially impact the company more than adjusting conditions to the individual when considering replacement could be 1.5x to 2x the original employees' salary or higher.

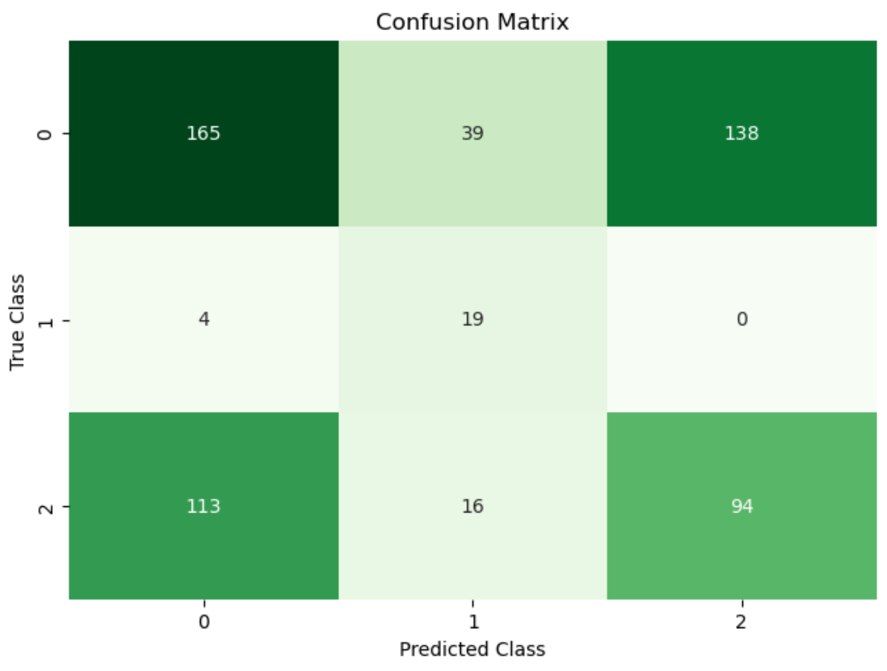


Figure 4 - Multinomial Logistic Regression Confusion Matrix

The confusion matrix from the multinomial logistic regression shows that 19/23 (82%) of class 1 employees and 94/223 (42%) of class 2 employees were correctly identified. The results show that most of the target classes should be predicted with this model to utilize resources the most effectively.

Considering the origin of the dataset was fictional, it should be fitted to other companies' employee records to validate goodness of fit. At the least, it should be validated on at least 2 previous employee engagement surveys before providing results. Although no personal identifiable information was available in the dataset, the recommendations from the model only be shared on a as needed basis. Only the employee ID will be presented in the recommendation to keep the analysis anonymous until action is taken by the human resources department.

References

Bersin, J. (2013, August 16). *Employee Retention Now a Big Issue: Why the Tide has Turned*. Retrieved from LinkedIn: https://www.linkedin.com/pulse/20130816200159-131079-employee-retention-now-a-big-issue-why-the-tide-has-turned/

Indeed. (2022, September 30). *What Is Employee Satisfaction and Why Is It Important?* Retrieved from https://ca.indeed.com/career-advice/career-development/what-is-employee-satisfaction

Sahoo, R. (2020). IBM Employee Dataset. Retrieved September 7, 2023, from https://www.kaggle.com/datasets/rohitsahoo/employee

Campbell, Julie_630_Project

November 9, 2023

[28]:
```
# Load libraries
import pandas as pd
import numpy as np
# Graphs
import matplotlib.pyplot as plt
import seaborn as sns
# Data Processing
from sklearn.preprocessing import OrdinalEncoder
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import chi2
# Test / Train
from sklearn.model_selection import train_test_split
# Model selection libraries
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler
# Model Search
from sklearn.model_selection import GridSearchCV
# Model Type
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
# Model Evaluation
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪roc_auc_score
# Disable warnings
import warnings
warnings.filterwarnings("ignore")
```

0.1 Import Files

[29]:
```
# Read csv file
employee_df = pd.read_csv('data/project/ibm_employee_train.csv')
```

```
[30]: # Combine test and train to resample
      employee_test = pd.read_csv('data/project/ibm_employee_test.csv')

      employee_df = employee_df.append(employee_test, ignore_index=True)
```

```
[4]: employee_df.shape
```

```
[4]: (1470, 35)
```

```
[31]: employee_df[['EmployeeCount','EmployeeNumber', 'Over18','StandardHours']].head()
```

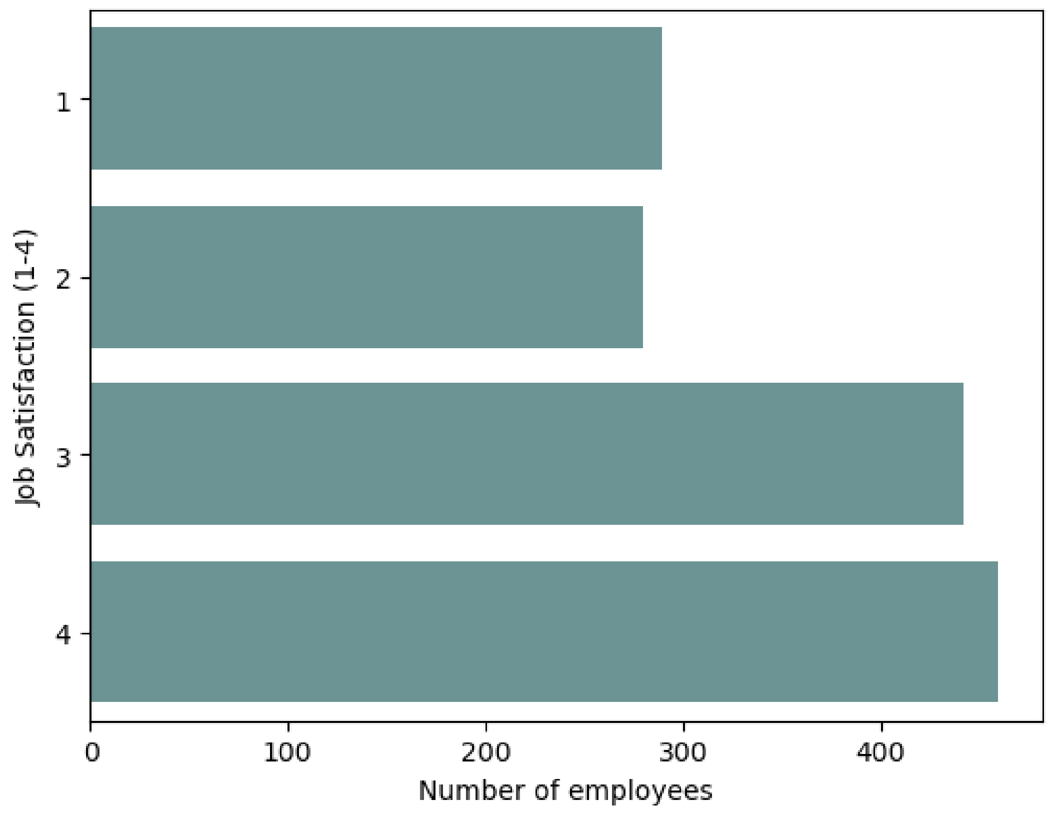
```
[31]:    EmployeeCount  EmployeeNumber Over18  StandardHours
      0              1               1      Y             80
      1              1               2      Y             80
      2              1               4      Y             80
      3              1               5      Y             80
      4              1               7      Y             80
```

```
[19]: employee_df[['EmployeeCount','EmployeeNumber', 'Over18','StandardHours']].
       ↪describe()
```

```
[19]:        EmployeeCount  EmployeeNumber  StandardHours
      count         1470.0     1470.000000         1470.0
      mean             1.0     1024.865306           80.0
      std              0.0      602.024335            0.0
      min              1.0        1.000000           80.0
      25%              1.0      491.250000           80.0
      50%              1.0     1020.500000           80.0
      75%              1.0     1555.750000           80.0
      max              1.0     2068.000000           80.0
```

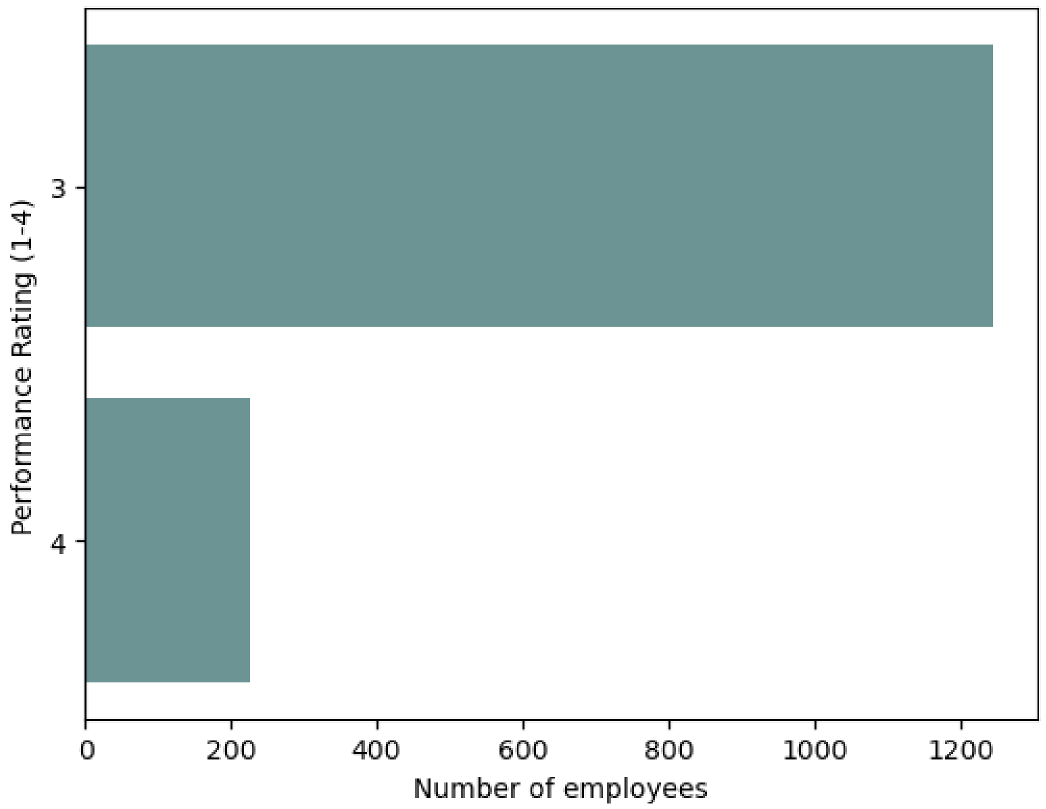
```
[636]: # View distribution of job satisfaction score
       sns.countplot(data=employee_df, y="JobSatisfaction", color='#679B9B')
       plt.xlabel('Number of employees')
       plt.ylabel('Job Satisfaction (1-4)')
```

```
[636]: Text(0, 0.5, 'Job Satisfaction (1-4)')
```



```
[637]: # View distribution of job satisfaction score
       sns.countplot(data=employee_df, y="PerformanceRating", color='#679B9B')
       plt.xlabel('Number of employees')
       plt.ylabel('Performance Rating (1-4)')
```

```
[637]: Text(0, 0.5, 'Performance Rating (1-4)')
```

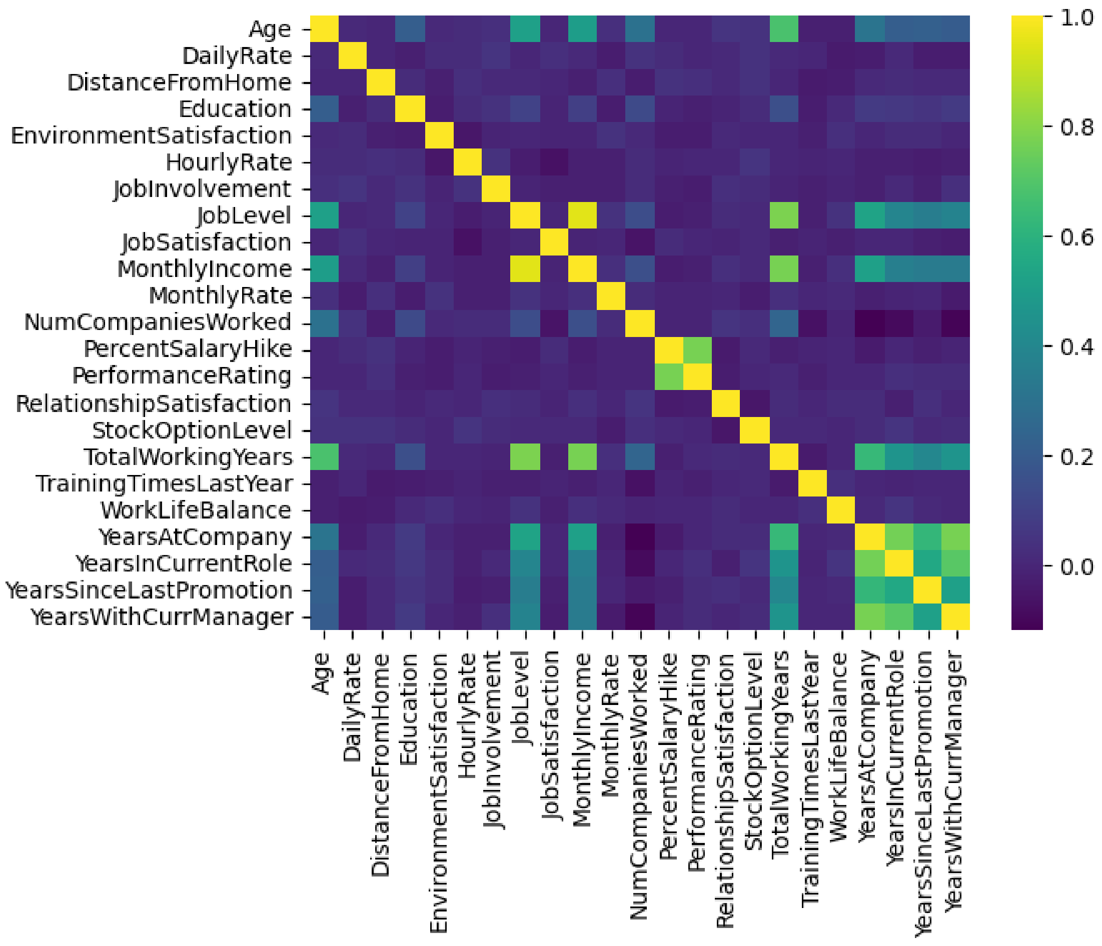


```
[4]: # Remove columns with one value or not consistent
     employee_df.
       ↪drop(['Attrition','EmployeeCount','EmployeeNumber','Over18','StandardHours'],␣
       ↪axis=1, inplace=True)
```

0.2 Correlation

```
[205]: matrix = employee_df.corr().round(2)
       sns.heatmap(matrix, cmap="viridis")
```

```
[205]: <AxesSubplot:>
```



[5]:
```
matrix = employee_df[['DailyRate', 'HourlyRate', 'MonthlyRate',␣
 ↪'MonthlyIncome']].corr().round(2)
matrix
```

[5]:
```
              DailyRate  HourlyRate  MonthlyRate  MonthlyIncome
DailyRate          1.00        0.02        -0.03           0.01
HourlyRate         0.02        1.00        -0.02          -0.02
MonthlyRate       -0.03       -0.02         1.00           0.03
MonthlyIncome      0.01       -0.02         0.03           1.00
```

[6]:
```
# Remove features with high correlation to another
employee_df.drop(['Age', 'YearsSinceLastPromotion', 'YearsInCurrentRole',␣
 ↪'YearsWithCurrManager'], axis=1, inplace=True)
```

[614]:
```
corr = employee_df.corrwith(employee_df.JobSatisfaction)
```

[615]:
```
print(corr.sort_values())
```

```
HourlyRate                  -0.071335
NumCompaniesWorked          -0.055699
JobInvolvement              -0.021476
TotalWorkingYears           -0.020185
WorkLifeBalance             -0.019459
RelationshipSatisfaction    -0.012454
Education                   -0.011296
MonthlyIncome               -0.007157
EnvironmentSatisfaction     -0.006784
TrainingTimesLastYear       -0.005779
YearsAtCompany              -0.003803
DistanceFromHome            -0.003669
JobLevel                    -0.001944
MonthlyRate                  0.000644
PerformanceRating            0.002297
StockOptionLevel             0.010690
PercentSalaryHike            0.020002
DailyRate                    0.030571
JobSatisfaction              1.000000
dtype: float64
```

0.3 Encoder

[8]:
```
# Seperate numeric and categorical features
df_numerical_features = employee_df.select_dtypes(include='number')
df_categorical_features = employee_df.select_dtypes(include='object')
```

[11]:
```
enc = OrdinalEncoder()
X_enc = enc.fit_transform(df_categorical_features)
```

[12]:
```
# Combine data
df_merged = pd.concat([df_numerical_features, df_categorical_features], axis=1)
```

[13]:
```
df_merged[['BusinessTravel', 'Department', 'EducationField', 'Gender',␣
 ↪'JobRole',
       'MaritalStatus', 'OverTime']] = pd.DataFrame(X_enc,␣
 ↪columns=['BusinessTravel', 'Department', 'EducationField', 'Gender',␣
 ↪'JobRole',
       'MaritalStatus', 'OverTime'])
```

[14]:
```
top_perform = df_merged["JobSatisfaction"][df_merged['PerformanceRating']>3].
 ↪replace({1:1, 2:2, 3:0, 4:0})
```

[15]:
```
mid_perfom = df_merged["JobSatisfaction"][df_merged['PerformanceRating']<=3].
 ↪replace({1:2, 2:2, 3:0, 4:0})
```

[16]:
```
df_merged["JobSatisfaction"] = pd.concat([top_perform, mid_perfom])
```

0.4 Split Test/Train

[17]:
```
# features
features = df_merged[['DailyRate', 'DistanceFromHome', 'Education',␣
 ↪'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel',
                      'MonthlyRate', 'PercentSalaryHike',␣
 ↪'RelationshipSatisfaction', 'StockOptionLevel', 'TotalWorkingYears',␣
 ↪'Gender',
                      'OverTime']]
# target
target = df_merged['JobSatisfaction']
```

[18]:
```
train_X, test_X, train_y, test_y = train_test_split(
    features, target, random_state=49, shuffle=True , test_size=0.4)
```

[19]:
```
sm = SMOTE(random_state=0)
X_res, y_res = sm.fit_resample(train_X,train_y)
```

0.5 Chi Square Test

[162]:
```
scores, pvalues = chi2(X_res, y_res)
pvalues=["{0:.7f}".format(x)for x in pvalues]
i = 0
for column in X_res:
    print(column + ": " + pvalues[i])
    i += 1
```

```
DailyRate: 0.0000000
DistanceFromHome: 0.0000000
Education: 0.0016198
EnvironmentSatisfaction: 0.0000354
HourlyRate: 0.0000000
JobInvolvement: 0.0011726
JobLevel: 0.0000353
MonthlyRate: 0.0000000
PercentSalaryHike: 0.0000000
RelationshipSatisfaction: 0.0000001
StockOptionLevel: 0.0000006
TotalWorkingYears: 0.0000000
Gender: 0.0239440
OverTime: 0.0044031
```

0.6 GridSearchCV

[20]:
```
# Create a base classifier
model = LogisticRegression(max_iter=1000, multi_class='multinomial')
```

[21]:
```
# Create a pipeline with a placeholder classifier
pipe = Pipeline([('scaler', MinMaxScaler(feature_range = (0, 1))),␣
 ↪('classifier', model)])
```

[22]:
```
# Check performance of logisitic regression
search_space = [{"classifier": [LogisticRegression()],
                 "classifier__multi_class": ['multinomial'],
                 "classifier__max_iter": [1000],
                 "classifier__C": [0.001, 0.01, 0.1, 1, 10, 100]}]
```

[172]:
```
# Check performance of random forest
search_space = [{"classifier": [RandomForestClassifier()],
                 "classifier__n_estimators": [100, 200, 300, 1000]}]
```

[178]:
```
# Create search space
search_space = [{"classifier": [KNeighborsClassifier()],
                 "classifier__n_neighbors": range(1,11),
                 "classifier__n_jobs": [-1]}]
```

[184]:
```
# Create search space
search_space = [{"classifier": [DecisionTreeClassifier()],
                 "classifier__criterion":['gini','entropy'],
                 "classifier__max_depth": np.arange(3, 15)}]
```

[191]:
```
search_space = [{"classifier": [XGBClassifier()],
                'classifier__max_depth': [3, 4, 5],
                'classifier__learning_rate': [0.1, 0.2, 0.3],
                'classifier__n_estimators': [50, 100, 150]
                }]
```

[23]:
```
# Create grid search with 3 folds using classifiers
classifier = GridSearchCV(pipe, search_space, cv=3, verbose=0)
```

0.6.1 Parameter Search

[24]:
```
# Fit data to best hyperparameters
classifier.fit(X_res, y_res)
```

[24]:
```
GridSearchCV(cv=3,
             estimator=Pipeline(steps=[('scaler', MinMaxScaler()),
                                       ('classifier',
                                        LogisticRegression(max_iter=1000,
multi_class='multinomial'))]),
             param_grid=[{'classifier': [LogisticRegression(C=100,
                                                            max_iter=1000,
multi_class='multinomial')],
                          'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100],
                          'classifier__max_iter': [1000],
                          'classifier__multi_class': ['multinomial']}])
```

```
[25]: # Best model parameters
      classifier.best_params_
```

```
[25]: {'classifier': LogisticRegression(C=100, max_iter=1000,
      multi_class='multinomial'),
       'classifier__C': 100,
       'classifier__max_iter': 1000,
       'classifier__multi_class': 'multinomial'}
```

```
[26]: classifier.best_score_
```

```
[26]: 0.6720333929636255
```

0.7 Model Evaluation

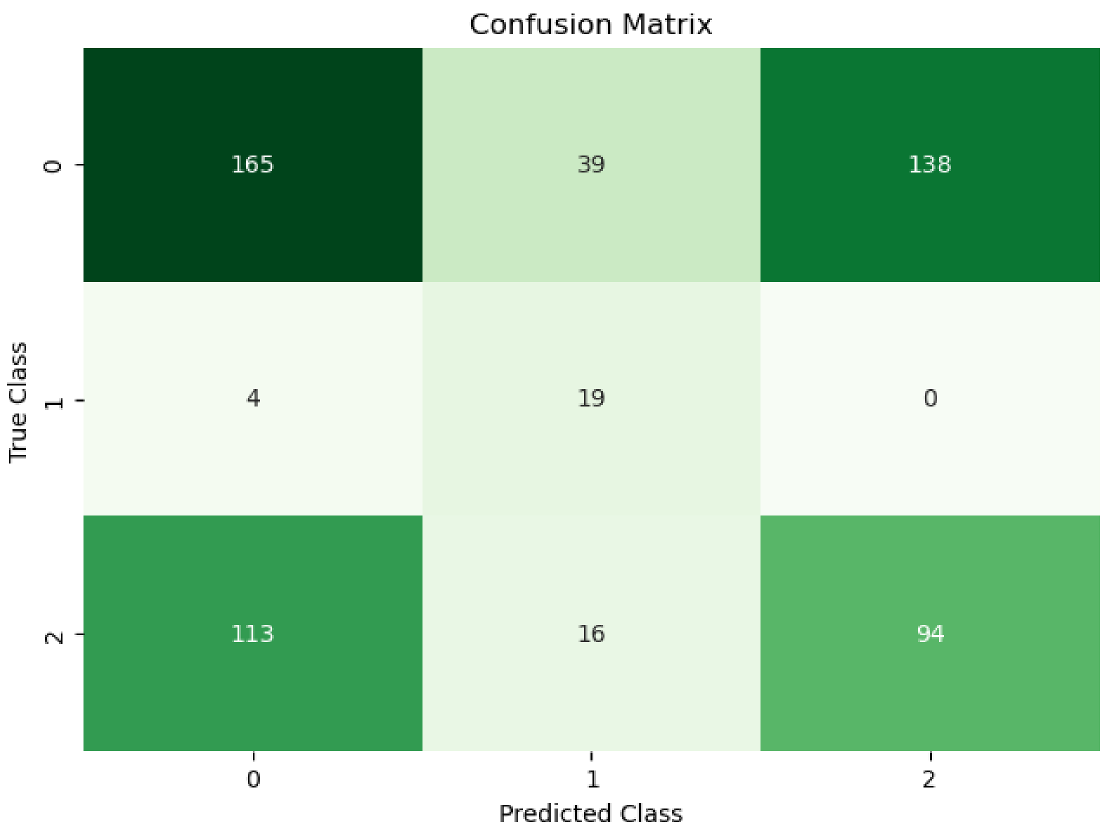
```
[170]: def ModelEvaluation(model):
           # Create a pipeline
           pipe = Pipeline([('scaler', MinMaxScaler(feature_range = (0, 1))),␣
        ↪('classifier', model)])

           # Fit SMOTE dataset to chosen model type
           pipe.fit(X_res, y_res)

           # Accuracy Score
           print('Accuracy Score')
           print(pipe.score(test_X, test_y), '\n')

           # Train model and make predictions
           target_predicted = pipe.predict(test_X)

           # Create confusion matrix
           matrix = confusion_matrix(test_y, target_predicted)

           # Relabel x and y labels
           # matrix_df = pd.DataFrame(matrix, range(1,5), range(1,5))

           # Plot heatmap of confusion matrix
           sns.heatmap(matrix, annot=True, cbar=None, cmap=plt.cm.Greens, fmt='g')
           plt.title("Confusion Matrix"), plt.tight_layout()
           plt.ylabel("True Class"), plt.xlabel("Predicted Class")
           plt.show()

           print('\n', classification_report(test_y, target_predicted), '\n\n')
```

```
    pred_prob = pipe.predict_proba(test_X)

    # 0.7  ROC < 0.8: Acceptable discrimination
    print('ROC AUC Score')
    print(roc_auc_score(test_y, pred_prob, multi_class='ovo'))
```

0.7.1 Multinomial Logistic Regression Model

[171]:
```
ModelEvaluation(LogisticRegression(C=100, max_iter=1000,␣
 ↪multi_class='multinomial'))
```

```
Accuracy Score
0.47278911564625853
```



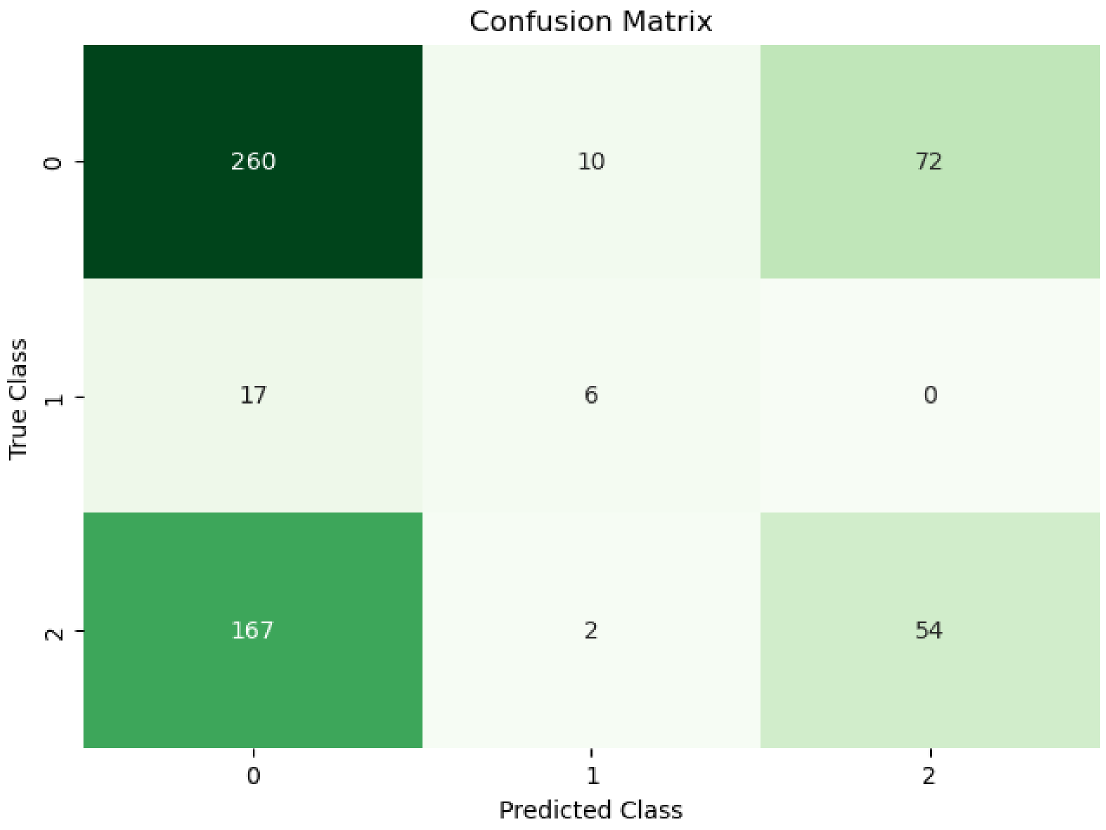
```
         precision     recall   f1-score    support

0           0.59       0.48       0.53        342
1           0.26       0.83       0.39         23
2           0.41       0.42       0.41        223
```

```
    accuracy                           0.47       588
   macro avg       0.42      0.58      0.44       588
weighted avg       0.50      0.47      0.48       588



ROC AUC Score
0.7712823956731151
```

0.7.2 Random Forest Model

[177]:
```
ModelEvaluation(RandomForestClassifier(n_estimators=300))
```

```
Accuracy Score
0.54421768707483
```

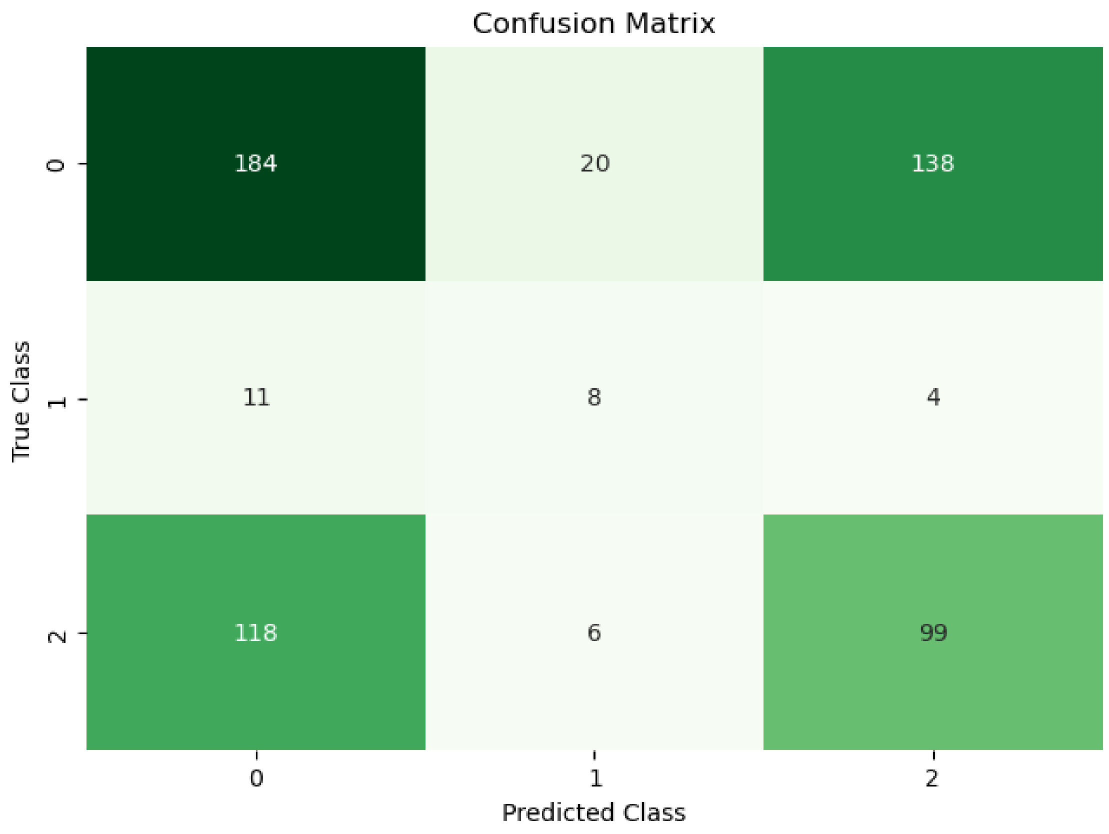


```
              precision    recall  f1-score   support

           0       0.59      0.76      0.66       342
           1       0.33      0.26      0.29        23
           2       0.43      0.24      0.31       223

    accuracy                           0.54       588
   macro avg       0.45      0.42      0.42       588
weighted avg       0.52      0.54      0.51       588



ROC AUC Score
0.759277217002125
```

0.7.3 K Neighbors Classifier

[183]:
```
ModelEvaluation(KNeighborsClassifier(n_jobs=-1, n_neighbors=1))
```

```
Accuracy Score
0.49489795918367346
```



```
              precision    recall  f1-score   support
```

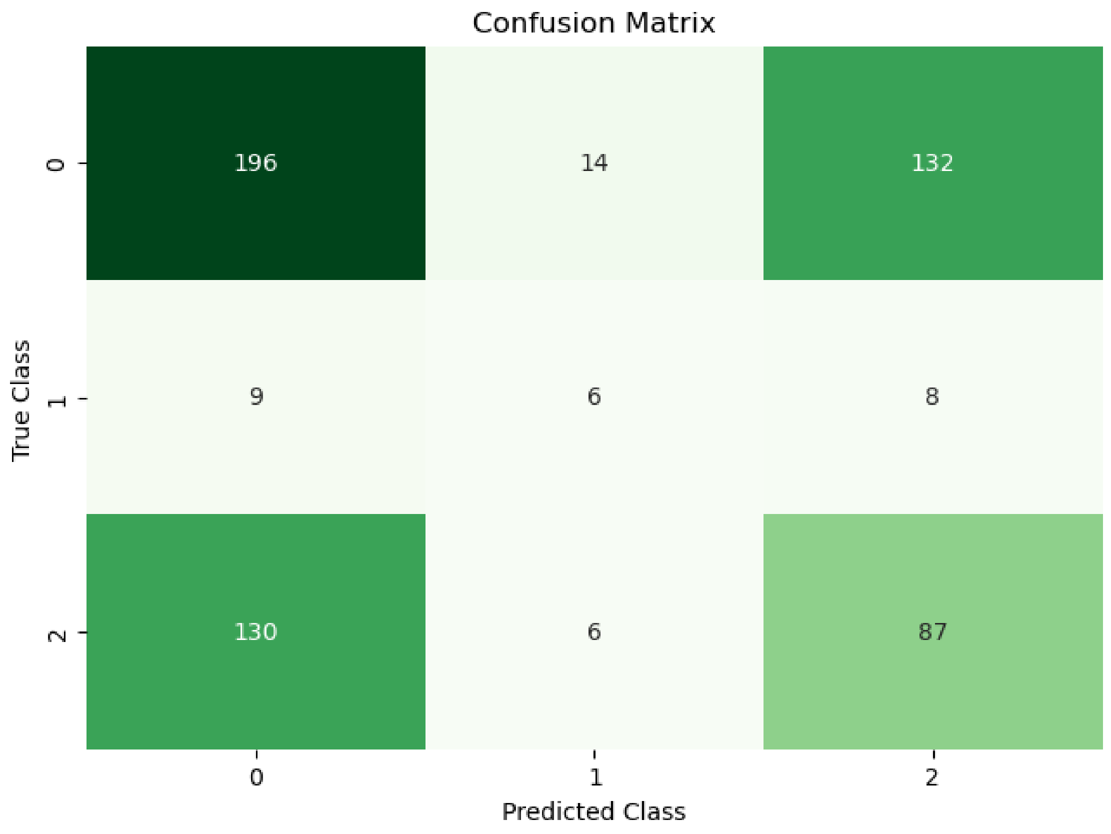
```
           0       0.59      0.54      0.56       342
           1       0.24      0.35      0.28        23
           2       0.41      0.44      0.43       223

    accuracy                           0.49       588
   macro avg       0.41      0.44      0.42       588
weighted avg       0.51      0.49      0.50       588
```

```
ROC AUC Score
0.5824459928009404
```

0.7.4 Decision Tree Classifier

[189]:
```
ModelEvaluation(DecisionTreeClassifier(criterion='gini', max_depth=12))
```

```
Accuracy Score
0.4914965986394558
```

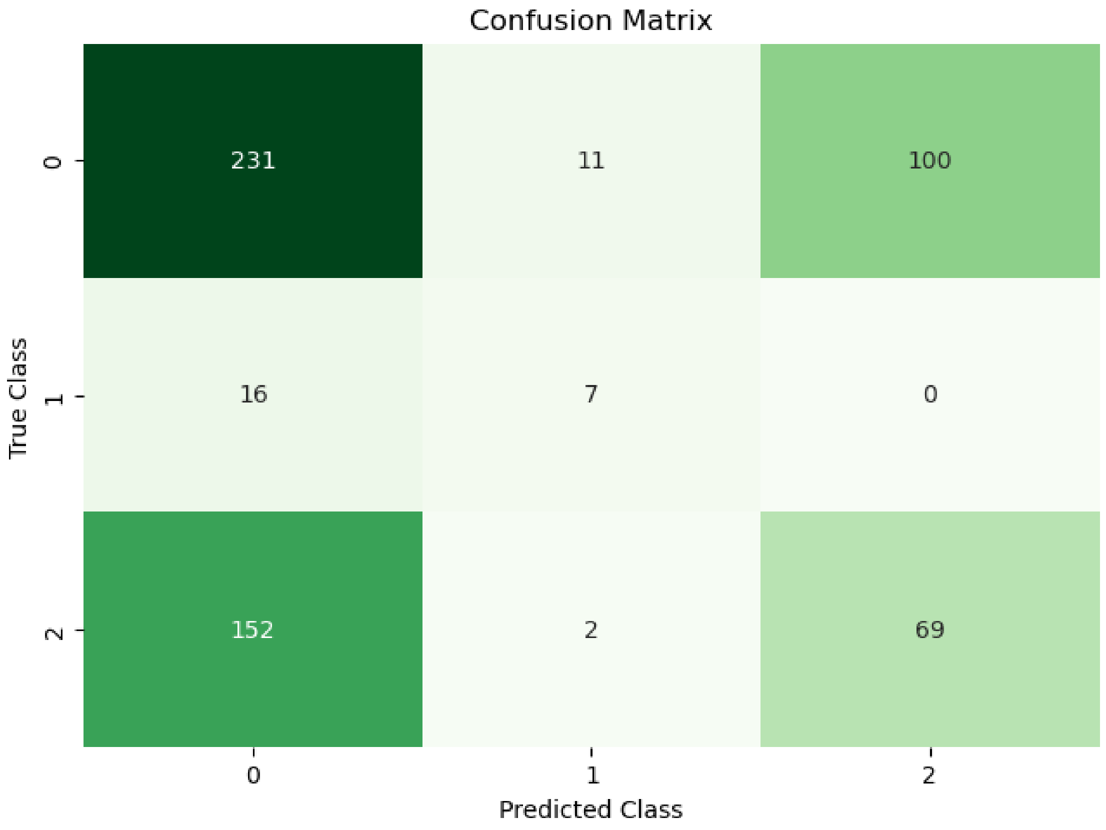


```
              precision    recall  f1-score   support

           0       0.59      0.57      0.58       342
           1       0.23      0.26      0.24        23
           2       0.38      0.39      0.39       223

    accuracy                           0.49       588
   macro avg       0.40      0.41      0.40       588
weighted avg       0.49      0.49      0.49       588



ROC AUC Score
0.5488648236131587
```

0.7.5 XGBoost Classifier

[196]:
```
ModelEvaluation(XGBClassifier(learning_rate=0.2, max_depth=5, n_estimators=150))
```

```
Accuracy Score
0.5221088435374149
```



```
              precision    recall  f1-score   support

           0       0.58      0.68      0.62       342
           1       0.35      0.30      0.33        23
           2       0.41      0.31      0.35       223

    accuracy                           0.52       588
   macro avg       0.45      0.43      0.43       588
weighted avg       0.51      0.52      0.51       588


ROC AUC Score
0.7050570904199907
```