Employee Attrition Prediction using Machine Learning Models

Kiran Reddy[1] **Hrudaya Jinna**[2] **Reshma Mamillapall**[3] **Deepthi Kondaveetir**[4]

University of Maryland Baltimore County
{RI99539,hrudayj1,r177,k289}@umbc.edu

Abstract

The term "Employee Attrition" describes when employees leave a company, either voluntarily or involuntarily. Due to the considerable influence, it could have on the organization's productivity and financial situation, it is one of the most urgent concerns that most businesses around the world are currently dealing with. As a result, companies must begin developing incredibly specific strategies to predict and avoid such a crisis, and it is in these circumstances that machine learning proves useful (to model the data and offer insights). This, together with the increased interest in machine learning for commercial decision-making, makes it necessary to go more into this area. Therefore, in this study, we have used Decision Trees, Support Vector Machine (SVM), XGBoost, and K Nearest neighbors algorithm to predict Employee Attrition. Through a thorough evaluation process, we choose the best model based on the accuracy of these models in predicting employee attrition.

1 Introduction

1.1 Problem Statement

While attrition is inevitable in every company, most businesses would benefit much from limiting it and being ready for circumstances that cannot be avoided. We can achieve this by creating a group of workers who are "at risk." Our first goal as researchers would be to analyze this data using classification algorithms and provide the essential insights to execute a segmentation of employees who are prone to attrition. The HR department has provided us with a wealth of data on employee demographics. In order to effectively anticipate the employees who are likely to attrition, this study primarily tries to predict the elements (Input variables / Features) that have a significant impact on attrition.

1.2 Project summary and methodology

To understand the distributions of all the features and their relationship with our target variable (Attrition) in this study, we first load the data and conduct exploratory data analysis utilizing correlation plots, histograms, bar graphs, etc. Once we have a complete understanding of our feature space, we start pre-processing the data in preparation for modeling. In this phase, we addition to principal component analysis, we have eliminated outliers using Box plots On normalized data to improve performance. We then developed classification models for attrition prediction using Decision Trees, Support Vector Machine (SVM), XGBoost, and K Nearest neighbor. Grid Search CV was used to adjust the f1 and hyperparameters for each of these models. In order to assess how well these models performed, the Accuracy, Precision, Recall, F1 Score, and Confusion matrix were utilized.

Steps performed in the project:

1. Recognize and comprehend the data (Numpy, Pandas)
2. Analyzing the data through exploratory data analysis (univariate and bi-variate analysis)
3. Data visualization to reveal relationships (Matplotlib and Seaborn)
4. Data pre-processing (Feature selection, outlier removal, PCA, and normalization)
5. Divide the dataset into two sections (Train data and Test data)
6. Selecting and using the train data set to train the selected machine learning model (Scikit Learn)
7. Fine-tuning each model's hyperparameters to increase prediction accuracy.
8. Using the Test set on the most accurate attrition prediction model
9. Analyzing each model's performance and selecting the top attrition prediction model.

1.3 Description of Dataset

We made use of General data set, Employee Survey Data set, Manager Survey Data set, In time Data set, Out time Data set and introduced a new column called employee id in all of these data sets which helped us to merge all these Data sets based on this new feature attribute. Data sets were collected from Kaggle website and it was challenging for us to get a proper data to satisfy the problem statement[1].

1.4 Target Variable

The Attrition column in this data set is the target feature. It is a category binary variable that accepts the values 0 and 1. Here, an employee's category "0" indicates that there is no attrition while category "1" indicates that there is a possibility of attrition.

1.5 Class descriptions for categorical attributes

Education = Below-College, College, Bachelor, Master, Doctor
Environment satisfaction = Low, Medium, High, Very High
Job involvement = Low, Medium, High, Very High
Job satisfaction = Low, Medium, High, Very High
Relationship satisfaction =Low, Medium, High, Very High
Performance rating = Low, Good, Excellent, Outstanding
Work life balance = Bad, Good, Better, Best

2 Methodology and Experiments

This project was developed using the Python programming language. The data sets used in this project used are in the form of a .csv file. We made use of all the necessary libraries like NumPy, pandas, matplotlib, seaborn , etc. We initially loaded all the data sets into the data frame, but we decided to have only one data frame holding one data set. So, we merged all these data sets by adding a new column called employee id to all these data sets, merged all these data sets, and created one final data set.

2.1 Data set

The below Figure 1 shows the data set columns and the data types of those columns.

2.2 Exploratory Data Analysis

Having imported our data and identified the variable types, we can now begin our inspection of each feature individually. In this section, we will investigate the correlation between each feature's distribution and our dependent variable (Attrition). We used histograms for numerical variables, count plots for categorical ones to visualize the relationships between numerical variables. Box plots

```
In [6]: final_df.dtypes

Out[6]: Age                         int64
        Attrition                  object
        BusinessTravel             object
        Department                 object
        DistanceFromHome            int64
        Education                   int64
        EducationField             object
        EmployeeCount               int64
        EmployeeID                  int64
        Gender                     object
        JobLevel                    int64
        JobRole                    object
        MaritalStatus              object
        MonthlyIncome               int64
        NumCompaniesWorked        float64
        Over18                     object
        PercentSalaryHike           int64
        StandardHours               int64
        StockOptionLevel            int64
        TotalWorkingYears         float64
        TrainingTimesLastYear       int64
        YearsAtCompany              int64
        YearsSinceLastPromotion     int64
        YearsWithCurrManager        int64
        EnvironmentSatisfaction   float64
        JobSatisfaction           float64
        WorkLifeBalance           float64
        JobInvolvement              int64
        PerformanceRating           int64
        dtype: object
```

Figure 1: Data Types of the data set features

Figure 2: Plotting all the columns with the unique values count

and sophisticated conditional plots were employed to identify associations between numeric and category variables. From the Figure 2 we plot the count plots of all the features in the final data set.

From the Figure 3 the key Observation from the plot are

1. Except-Age, most of the columns follow a skew distribution, whereas Age's feature distribution is nearly almost a normal Distribution.

2. Since it is not necessary for the independent variables to have a normal distribution when using logistic regression. Therefore, we are not making any changes to the distribution of features that are asymmetrical to the normal Distribution.

In the Figure 4,5,6 we plot the columns with respect to Attrition which is YES/NO.

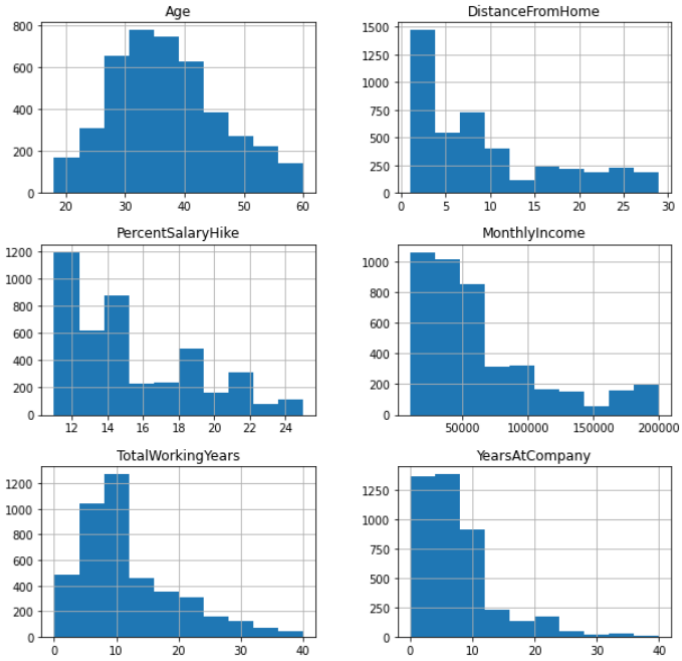

Figure 3: Plotting numerical columns with respect to count

2.2.1 Outlier Identification

The outliers that exist in our data may be due to natural occurrences, mistakes made during data entry, or inaccurate measurements. Because the majority of machine learning algorithms are sensitive to outliers, it is preferable that we remove such outliers before we begin developing our ML models. All of these outliers raise the variance in our data and lower the power of statistical tests. Visualizing the box plot distribution of each feature is an effective method for locating and eliminating outliers in a data set[2]. Based on the box plot shown in Figure 7, we are checking if there are any outliers in the numerical column.We can observe outliers on Monthly Income, Total Working Years, and Years At Company Columns. However,we can observe that some of the features are not outliers. This is due to the fact that there is a high probability of those numerical values occurring on those features or columns. So we are not removing outliers on our data set.

2.2.2 Correlation Plot

We have analyzed the relationship between features by using the correlation plots that is shown in Figure 8. This is possible because we now have a comprehensive understanding of all of the variables, which we will refer to as features, as well as their respective distributions. When we have dependent variables, changes to one variable can induce a swing of our output throughout the model fitting process. This is something that we want to avoid at all costs. Therefore, locating these kinds of correlations and dealing with them during the data pre- processing stage is of the utmost importance to the process of making our classification model more accurate.

2.3 Data Preprocessing

Steps performed in Data preprocessing

Figure 4: Plotting columns with respect to Attrition YES/NO

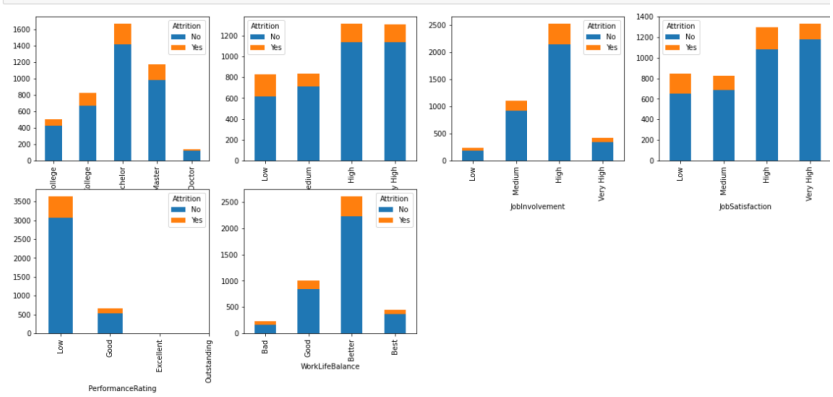

Figure 5: Categorical columns with respect to Attrition YES/NO

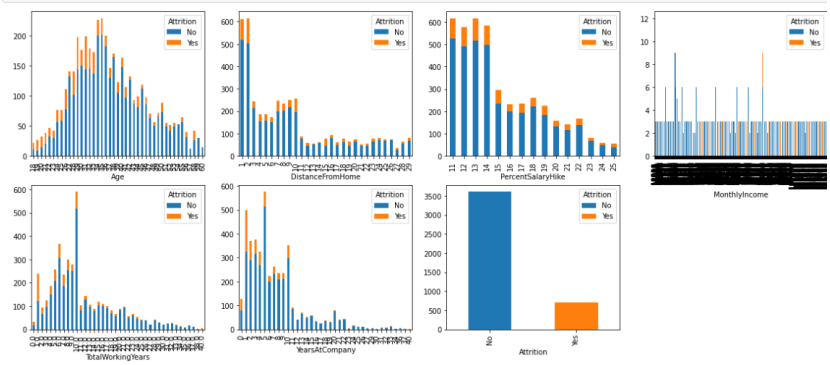

Figure 6: Columns with respect to Attrition YES/NO

1. Merging the data sets:- We have merged all the data sets and build one final data set. To double check if the data sets are merged, we have written a code which equates the length of final data sets to the addition done on all the individual data set length. If both LHS and RHS are equated then we print as 'OK'. The code snippet for this step is in the Figure 9.
2. Finding out the Null values:- Written a code to find out null values and found that there are 111 null values. Hence, we deleted all those null value rows and double checked if there are any null values. We found no null values.The code for this step is shown in the Figure 10.
3. Checking for Duplicate rows:- Written a code to check for duplicate rows. We found that there are no duplicate rows in the dataset and the code for this step is shown in the Figure 11 .
4. Applying Dummy encoding to the data set:- Encoding categorical variables can be accomplished in a couple of different ways. Say, one categorical variable has n values. In one-hot encoding, it is converted into n variables, whereas in dummy encoding, it is converted into n variables plus one. Consider the following scenario: there are k categorical variables, and each one has n possible values. The output of a single hot encoding is kn variables, but the output of a dummy encoding is kn-k variables[3].
5. Label Encoding:- We have made use of label encoding to convert all the text data to numerical data in the data set.

3 Model Building

We have built the model by considering few algorithms like Decision trees, Support Vector Machine, K Nearest neighbors, XGBoost and Convolutional Neural Network.

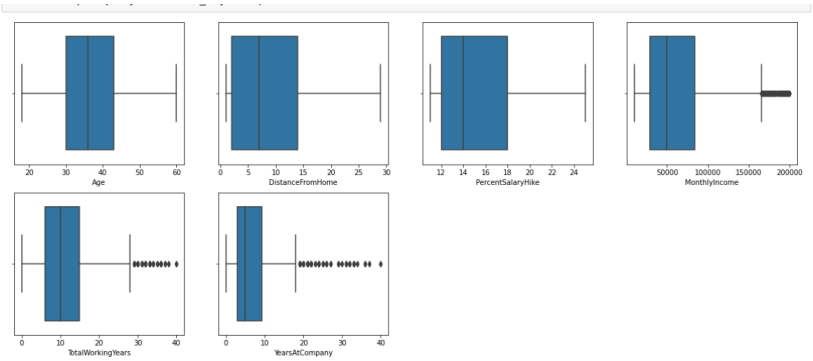

Figure 7: Detection of Outliers for Numerical columns

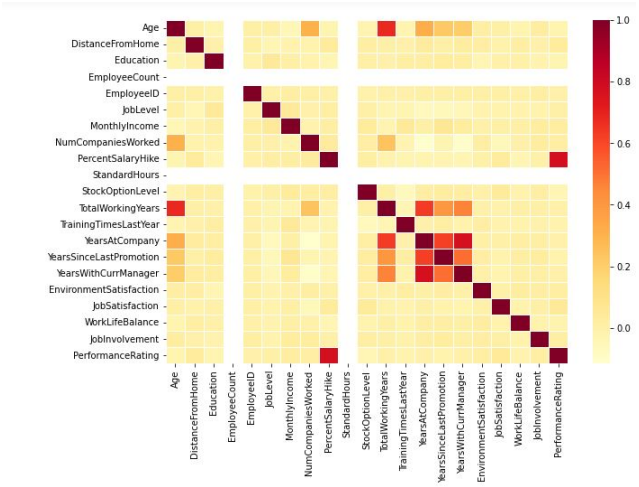

Figure 8: Finding the correlation between all the features

```
generalData_df      = pd.read_csv("E:/ML/Final_Project/Code/general_data.csv")
employee_survey_df  = pd.read_csv("E:/ML/Final_Project/Code/employee_survey_data.csv")
manager_survey_df   = pd.read_csv("E:/ML/Final_Project/Code/manager_survey_data.csv")
intime_df           = pd.read_csv("E:/ML/Final_Project/Code/in_time.csv")
outtime_df          = pd.read_csv("E:/ML/Final_Project/Code/out_time.csv")
```

```
df1=pd.merge(generalData_df,employee_survey_df,on='EmployeeID')
final_df=pd.merge(df1,manager_survey_df,on='EmployeeID')

if(len(final_df.columns.tolist())   == (len(generalData_df.columns.tolist()) +
                                        len(employee_survey_df.columns.tolist()) +
                                        len(manager_survey_df.columns.tolist())) - 2):
  print("OK")
```

```
OK
```

Figure 9: Merging the data sets and double checking with code logic

```
print(final_df.isnull().sum().sum())
final_df.dropna(inplace=True)
print(final_df.isnull().any().sum())
```

```
111
0
```

Figure 10: Code snippet for finding out the Null values

3.1 Decision Trees

The technique of supervised learning known as the decision tree can be applied to problems involving both classification and regression, but in practice, it is most often utilized for the resolution of

Checking for duplicates

```
final_df_dupliactedRows=final_df[final_df.duplicated( keep=False)].shape[0]
if(final_df_dupliactedRows==0):
  print("there is No duplicate elements in Final_df")
else:
  print("there is  duplicate elements in Final_df")
```

```
there is No duplicate elements in Final_df
```

Figure 11: Code Snipped for checking for Duplicate rows

```
#foe i in [X_train,X_cv,X_test]:
X_train=pd.get_dummies(X_train)
X_test=pd.get_dummies(X_test)
```

Figure 12: Dummy Encoding

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
final_df["BusinessTravel"] = le.fit_transform(final_df["BusinessTravel"])
final_df["Department"] = le.fit_transform(final_df["Department"])
final_df["EducationField"] = le.fit_transform(final_df["EducationField"])
final_df["Gender"] = le.fit_transform(final_df["Gender"])
#final_df["Attrition"] = le.fit_transform(final_df["Attrition"])
final_df["JobRole"] = le.fit_transform(final_df["JobRole"])
final_df["MaritalStatus"] = le.fit_transform(final_df["MaritalStatus"])
final_df["Over18"] = le.fit_transform(final_df["Over18"])
```

Figure 13: Label Encoding

classification issues. It is a classifier in the form of a tree, with internal nodes representing the features of a dataset, branches representing the decision rules, and each leaf node representing the conclusion of the classification. The Decision Node[4] and the Leaf Node are the two different types of nodes that can be found in a Decision tree. Leaf nodes are the output of those decisions and do not contain any additional branches, whereas Decision nodes are used to make any decision and have many branches. Leaf nodes can be found in a decision tree. The decisions or the test are carried out based on the characteristics of the dataset that has been provided.

Classifiers known as decision trees are constructed in the form of hierarchical data structures utilizing the divide and conquer method. In the decision tree, determining the class of a sample is as easy as moving from the tree's root node to any of its leaf nodes. The leaf nodes are the ones that provide the labels for the various classes. Now that we have our data set, we can create several decision trees to represent the data. Since we have this option, our primary objective will be to select the decision tree that provides the greatest representation of our data so that we can learn from it. When working on this project, we constructed our classification model by using the tree library provided by sklearn. The final classification model achieves an accuracy of 98 percentage and an F1-score of 0.95 across the board for positive and negative classes respectively. Excellent performance was achieved by the model that was constructed, and it is now ready to be put into use to forecast future employee turnover. During EDA, we uncovered a lot of new information regarding the turnover of employees. Employee turnover can be driven by a number of factors, including human resource work, an unhealthy balance between work and leisure, frequent travel, and employees who aren't married. Therefore, the organization should focus on these issues in order to minimise employee turnover.

```
accuracy_score(y_test, y_pred)
```

```
0.9848837209302326
```

```
precision_score(y_test, y_pred)
```

```
0.95
```

```
recall_score(y_test, y_pred)
```

```
0.9568345323741008
```

```
f1_score(y_test, y_pred)
```

```
0.953405017921147
```

```
from sklearn.metrics import classification_report
```

```
cr = classification_report(y_test, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       721
           1       0.95      0.96      0.95       139

    accuracy                           0.98       860
   macro avg       0.97      0.97      0.97       860
weighted avg       0.98      0.98      0.98       860
```

Figure 14: Decision Tree Performance

3.2 K Nearest neighbors

One of the most fundamental machine learning algorithms, K-Nearest Neighbor makes use of the Supervised Learning approach to data analysis. The K-Nearest Neighbors algorithm[5] makes the assumption that the new case or data is comparable to existing cases and places the new example into the category that is the most similar to the categories that are already accessible. The KNN algorithm remembers all of the accessible data and determines how to categorize a new data point depending on how similar it is to the stored data. This indicates that when fresh data becomes available, it may be quickly sorted into a well-suited category by making use of the KNN method. The K-Nearest Neighbors approach can be used for classification as well as regression, although the majority of the time, it is employed for the classification problems. The K-Nearest Neighbors technique is non-parametric, which implies that it does not make any assumptions about the data it is analyzing. It is also known as a lazy learner algorithm because rather than immediately learning from the training set, it saves the data set and then, when the time comes for classification, it executes an action on the data set. During the training phase, the KNN algorithm does nothing more than store the information. When it receives new data, the system then classifies that data into a category that is quite similar to the new data. This algorithm worked well for our problem statement. The Accuracy score was around 83.83% for K value of 20. This algorithm has very decent performance to our data set.

3.3 Support Vector machine

Support Vector Machine, or SVM for short, is a technique for supervised machine learning that can be used for classification as well as regression. Despite the fact that we also discuss regression

```
for keys, values in scores.items():
    print(keys, ':', values)
```

```
2 : [0.8627906976744186, 0.8162790697674419]
3 : [0.8593023255813953, 0.7779069767441861]
4 : [0.8453488372093023, 0.8302325581395349]
5 : [0.8441860465116279, 0.8127906976744186]
6 : [0.8404069767441861, 0.8325581395348837]
7 : [0.8421511627906977, 0.8267441860465117]
8 : [0.8401162790697675, 0.8383720930232558]
9 : [0.8404069767441861, 0.8348837209302326]
10 : [0.8395348837209302, 0.8383720930232558]
11 : [0.8415697674418605, 0.8383720930232558]
12 : [0.8392441860465116, 0.8395348837209302]
13 : [0.8395348837209302, 0.8383720930232558]
14 : [0.8392441860465116, 0.8395348837209302]
15 : [0.8392441860465116, 0.8395348837209302]
16 : [0.8386627906976745, 0.8395348837209302]
17 : [0.8398255813953488, 0.8383720930232558]
18 : [0.8386627906976745, 0.8395348837209302]
19 : [0.838953488372093, 0.8395348837209302]
20 : [0.8383720930232558, 0.8383720930232558]
```

Figure 15: K Nearest neighbors training accuracy and testing accuracy

```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

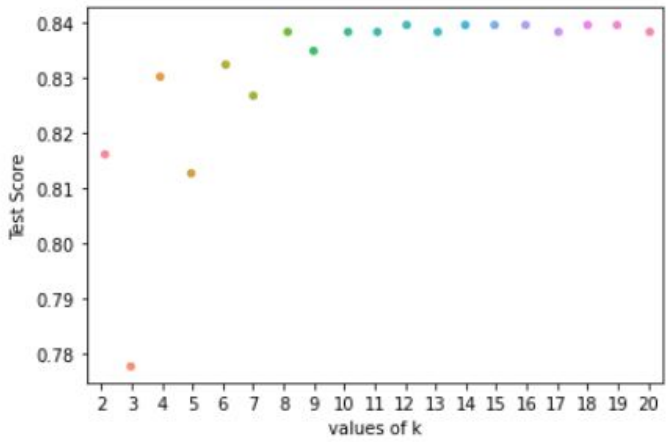

Figure 16: Plotting test accuracy for K Nearest neighbors

problems, its use is more appropriate for classification. The goal of the Support Vector Machine (SVM) algorithm[6] is to locate a hyperplane in an N-dimensional space that can classify the data points in a separate manner. The number of features determines which dimension of the hyperplane will be used. If there are only two input features, then the hyperplane is nothing more than a straight line. If there are three features used as input, then the hyperplane transforms into a two-dimensional

plane. When there are more than three aspects, it is more difficult to visualize what the whole thing might be like.

SVM is applied when the instances and classes in question cannot be linearly separated. In addition to doing linear classification, support vector machines (SVMs) are also capable of performing non-linear classification in an effective manner by employing a technique known as the kernel trick, which maps their inputs implicitly into high-dimensional feature spaces. The support vector machine (SVM) assigns training examples to points in space in such a way as to maximize the difference in size that exists between the two categories. After then, fresh instances are mapped into the same space, and a prediction is made regarding which category they belong to based on which side of the gap they fall on.

In the Support Vector Machine approach, we produce a hyperplane in the feature space that divides data into classes. This hyperplane can be thought of as a partition. It works on maximizing the margin up to the point that is the closest. One of the primary factors that led us to select SVM for our data set was the efficiency with which it performed in feature spaces with a greater dimension. Our Soft margin Support vector machine classification model was trained for this project using a variety of different penalty terms (C) and Gamma values for rbf and polynomial kernels. As observed in Fig. The best SVM classification model for our data set, which provides us with an accuracy of prediction of 83.83 percent. In the Figure 17 we can observe the Accuracy of SVM implementation.

```
testaccuracy_clf = accuracy_score(y_test, predict_test_clf)
print('accuracy_score on test dataset : ', testaccuracy_clf)
```

```
accuracy_score on test dataset :  0.8383720930232558
```

Figure 17: Support Vector Machine Accuracy

3.4 XGBoost

The Gradient Boosted decision trees algorithm can be implemented with XGBoost. XGBoost models have an overwhelming advantage in the majority of Kaggle Competitions.

The formation of decision trees takes place in a step-by-step fashion within this method. Weights are an essential component of the XGBoost algorithm[7]. All of the independent variables are each given a weight, and those weights are then taken into account in the decision tree that is used to predict the results. The weight of factors whose outcomes were incorrectly anticipated by the tree is increased, and these outcomes are then placed into a second decision tree. When combined, the results of these separate classifiers and predictors produce a robust and accurate model. It is able to work on problems involving regression, classification, ranking, and user-defined prediction issues.

We found that this algorithm can be tried for this problem statement. The performance of this algorithm for our data set and problem statement worked excellent. The overall test accuracy was 99.65% and test data sensitivity was 97.84% and test data specificity happens to be 100%. In the Figure 18 The performance of XGBoost is shown.

3.5 Convolutional Neural Network

A Convolutional Neural Network, often known as a CNN, is made up of one or more convolutional layers (typically followed by a sub sampling step), which are then followed by one or more fully connected layers, just like a regular multilayer neural network would be. The architecture of a CNN is meant to work in conjunction with the 2D structure of the images that it receives as input (or other 2D input such as a speech signal). This is accomplished through the use of local connections and tied weights, and is then followed up with a form of pooling that produces features that are translation invariant. Another advantage of convolutional neural networks (CNNs) is that they are simpler to train and have a significantly reduced number of parameters in comparison to fully connected networks that have the same number of hidden units. In this article, we will talk about the architecture of a CNN as well as the back propagation technique that is used to compute the gradient with respect

```
# Let's see the sensitivity of our model
testsensitivity= TP / float(TP+FN)
testsensitivity
```

```
0.9784172661870504
```

```
# Let us calculate specificity
testspecificity= TN / float(TN+FP)
testspecificity
```

```
1.0
```

```
# Calculate false postive rate - predicting Attrition when customer does not have Attrited
print(FP/ float(TN+FP))
```

```
0.0
```

```
# Positive predictive value
print (TP / float(TP+FP))
```

```
1.0
```

```
from sklearn.metrics import precision_score, recall_score
precision_score(y_test, predict_test)
```

```
1.0
```

```
recall_score(y_test, predict_test)
```

```
0.9784172661870504
```

```
print("Test Data Accuracy     :{} %".format(round((testaccuracy*100),2)))
print("Test Data Sensitivity  :{} %".format(round((testsensitivity*100),2)))
print("Test Data Specificity  :{} %".format(round((testspecificity*100),2)))
```

```
Test Data Accuracy     :99.65 %
Test Data Sensitivity  :97.84 %
Test Data Specificity  :100.0 %
```

Figure 18: XGBoost performance

to the parameters of the model in order to make use of gradient-based optimization. Please refer to the individual tutorials on convolution and pooling for further information regarding these particular procedures.

We have made use of RELU and Softmax activation functions in this algorithm. We utilized Adam's optimizer for optimizing. We have built this model by using Tensor Flow. Dense and Dropout was also utilized for layers in model. In total, we have executed 10 epochs for this code. The accuracy obtained for this algorithm was 85.11%. In the Figure 19 implementation of CNN epoch with validation loss and accuracy is shown.In the Figure 20 No of layers and parameters have shown in implementing CNN in our project.Accuracy of CNN is shown in FIgure 20

4 Conclusion and Future Work

We found that all these algorithms such as Decision Trees, K Nearest Neighbors, Convolutional Neural Network,XGBoost and Support Vector Machine worked good with the data set. But XGBoost and Decision tree worked excellent and they do have accuracy sores of around 99.65% and 98.4% respectively. Few major causes of employee attrition includes human resource work, bad work life balance, frequent travels and unmarried employees. So, in order to reduce attrition company should focus on these reasons. All these attributes or features contribute more towards employee attrition. In the table 1 the accuracy of different algorithms is shown. Even though we have been successful in predicting and developing a strategic retention plan for employees who are likely to leave their jobs, the predictions that are based on our data still have room for improvement. A few of these scenarios are ones in which we believe that our project has the most potential for future growth. These include the following: making our data set more balanced by increasing the number of samples of employees who are prone to attrition (Design of a better balanced data set), implementation of better risk buckets, and strategic retention plans. Additionally, applying our classification models by making use of deep neural networks is one of the most viable regions in this research, and it has a significant amount of expansion potential in the near and distant futures.

```
model.fit(X_train,y_train,validation_data=(X_test,y_test),batch_size=20,epochs=10,verbose=1)

Epoch 1/10
172/172 [==============================] - 5s 8ms/step - loss: 0.4552 - accuracy: 0.8328 - val_loss: 0.4131 - val_accuracy:
0.8465
Epoch 2/10
172/172 [==============================] - 1s 4ms/step - loss: 0.4228 - accuracy: 0.8363 - val_loss: 0.3940 - val_accuracy:
0.8465
Epoch 3/10
172/172 [==============================] - 1s 4ms/step - loss: 0.4072 - accuracy: 0.8363 - val_loss: 0.3862 - val_accuracy:
0.8465
Epoch 4/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3930 - accuracy: 0.8410 - val_loss: 0.3841 - val_accuracy:
0.8453
Epoch 5/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3933 - accuracy: 0.8448 - val_loss: 0.3777 - val_accuracy:
0.8500
Epoch 6/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3884 - accuracy: 0.8477 - val_loss: 0.3952 - val_accuracy:
0.8477
Epoch 7/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3903 - accuracy: 0.8459 - val_loss: 0.4068 - val_accuracy:
0.8570
Epoch 8/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3811 - accuracy: 0.8500 - val_loss: 0.3945 - val_accuracy:
0.8384
Epoch 9/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3887 - accuracy: 0.8404 - val_loss: 0.3844 - val_accuracy:
0.8488
Epoch 10/10
172/172 [==============================] - 1s 4ms/step - loss: 0.3795 - accuracy: 0.8433 - val_loss: 0.3903 - val_accuracy:
0.8512
```

Figure 19: CNN epoch with validation loss and accuracy

```
model=Sequential()
model.add(Dense(1000,input_dim=28,activation='relu'))
model.add(Dense(500,activation='relu'))
model.add(Dense(300,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 1000)              29000
_________________________________________________________________
dense_5 (Dense)              (None, 500)               500500
_________________________________________________________________
dense_6 (Dense)              (None, 300)               150300
_________________________________________________________________
dropout_1 (Dropout)          (None, 300)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 602
=================================================================
Total params: 680,402
Trainable params: 680,402
Non-trainable params: 0
_________________________________________________________________
```

Figure 20: Summary of CNN Layers and Parameters

```
prediction=model.predict(X_test)
length=len(prediction)
y_label=np.argmax(y_test,axis=1)
predict_label=np.argmax(prediction,axis=1)

accuracy=np.sum(y_label==predict_label)/length * 100
print("Accuracy of the dataset",accuracy )
```

```
Accuracy of the dataset 85.11627906976744
```

Figure 21: Accuracy of the CNN

Algorithms	**Accuracy Score**
Decision Tree	98.4
K Nearest Neighbor	83.8
Support Vector Machine	83.8
Xgboost	99.65
Convolution Neural Network	85.11

Table 1: The different Algorithm Implementation and Accuracy Scores

5 Novelty

1. We Initially considered IBM data set for detecting Employee Attrition. But that data set have very less number of rows. So we considered four different data sets and induced new column called employee in all these data sets so as to merge them into one final data set.
2. We initially made use of one hot encoding for converting text data to numerical data but the features in the data set was getting split. Hence, we did'nt wanted to increase the features as we have enough. We decided to drop one hot encoding and made use of label encoding.
3. We have found out the Outliers in the data set for the selected features and thus decided upon feature selection.
4. In Exploratory Data Analysis, we have learnt a lot about the data behaviour and how each and every feature is correlated with each other. By looking into thing in details, we were able to find the best suited features required to predict the target feature i.e. Attrition.
5. We Initially though Decision tree would provide good accuracy but after implementing different algorithms like K Nearest Neighbors, Convolutional Neural Network, Support Vector Machine and XGBoost, we found that surprisingly XGBoost algorithm worked pretty good with this data set for this problem statement.

6 References

[1]Employee Turnover Prediction with Machine Learning: A Reliable Approach Yue Zhao1() , Maciej K. Hryniewicki2 , Francesca Cheng2, Boyang Fu3, and Xiaoyu Zhu4

[2]N. Bhartiya, S. Jannu, P. Shukla and R. Chapaneri, "Employee Attrition Prediction Using Classification Models," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019, pp. 1-6, doi: 10.1109/I2CT45611.2019.9033784.

[3] S. S. Alduayj and K. Rajpoot, "Predicting Employee Attrition using Machine Learning," 2018 International Conference on Innovations in Information Technology (IIT), 2018, pp. 93-98, doi: 10.1109/INNOVATIONS.2018.8605976.

[4] P. Sadana and D. Munnuru, "Machine Learning Model to Predict WorkForce Attrition," 2021 6th International Conference for Convergence in Technology (I2CT), 2021, pp. 1-6, doi: 10.1109/I2CT51068.2021.9418140.

[5]S. Krishna and S. Sidharth, "Analyzing Employee Attrition Using Machine Learning: the New AI Approach," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1-14, doi: 10.1109/I2CT54291.2022.9825342.

[6] Raza, Ali. (2022). Predicting Employee Attrition Using Machine Learning Approaches. Applied Sciences. 12. 10.3390/app12136424.

[7] M. Subhashini,R. Gopinath, EMPLOYEE ATTRITION PREDICTION IN INDUSTRY USING MACHINE LEARNING TECHNIQUES, International Journal of Advanced Research in Engineering and Technology (IJARET), 2020, 11(12), PP3329-3341.