For the final project, we need to build a model to predict whether a person will remain in a hypothetical trade union, the United Data Scientists Union (UDSU). Assume a country with a major trade union for data scientists. Each member pays some dues that amount to it each month and gets various benefits such as representation with employers and continuing education. The project can be split into three parts: data cleaning, correlation and PCA, and then finally feature selection and modeling.

1. **Data Cleaning**

The first thing to do is check for NaN values in the training data. For the training data given, there were no missing values. Then, we can look at the feature types (float, int, object) to see if there are numerical features that are of type ‘object’. This will indicate that the variables are strings. Looking at the feature types, TotalDues is considered an object so that variable was converted to a float type. After, resulting NaN values were replaced with the median value.

Third, looking at only MonthsInUnion, MonthlyDues and TotalDues (dropping Management), we want to look at outliers in continuous variables. We notice that the 3rd quartiles of MonthlyDues and MonthsInUnion are way smaller than the maximum values. This might suggest the existence of outliers. To check this, we can create a boxplot.

Table

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Based on the boxplot, there are indeed outliers that impair our models. In this case we should remove data less than 2000 in the MonthlyDues and less than 500 in MonthsInUnion.

Lastly, we can perform the conversion to numeric and standardization. This consists of using the function LabelEncoder to perform encoding for binary variables. As for the multiclass ones we can use One-Hot Encoding. We can also perform get\_dummies for categorical variables and StandardScaler to standardize the float variables.

Graphical user interface, text, application, email

Description automatically generated

Then, our data will look like this after cleaning up:

Table

Description automatically generated

1. **PCA and Correlation**

In this section, we will first plot scatter plots for continuous variables. Plotting scatter plots allows us to see if there are any patterns between the three continuous variables. Based on the scatter plot below, there are no obvious patterns.

A picture containing diagram

Description automatically generated

Then, we will calculate the Pearson correlation coefficient between the variables, the target, as well as the p-values. Finding the correlation coefficients allows us to find how strong a relationship is between data. The output will return a value between -1 and 1 where: 1 indicates a strong positive relationship, -1 indicates a strong negative relationship and 0 indicates no relationship at all. From the Pearson correlation matrix, we can see that there is not a single variable that is highly correlated with the target. Later we will perform Recursive Feature Elimination with the modeling section so that we can consider smaller sets of features.

A picture containing graphical user interface

Description automatically generated

Finally, we will apply PCA to the data and plot the Scree Plot. The scree plot shows us that most of the variance can be explained with the two first principal components. We tried training the models with them (and adding more later), but it gave us results that were either less or as much as the features. In this case it was better to just go with the current features and perform feature selection.

Chart, line chart

Description automatically generated

1. **Feature Selection and Modeling**

We will use four models in our modeling phase: Logistic Regression, SVM with a polynomial kernel, Decision Tree and Random Forest. To select features, we will use Recursive Feature Elimination (RFE), which is a wrapper-type feature selection algorithm. RFE starts with all the features in the dataset and fitting a ML algorithm, then ranking features by importance, removing the least important features and re-fitting the model until it reaches a desired number.

We will use RFE with Logistic Regression to extract optimal features, then we will use them to fit the same model, as well as SVM. We will use it again with Decision Tree to extract other optimal features and then fit it. And finally do the same for Random Forest.

We will also optimize the models by the following:

* Finding the best polynomial degree for the SVM
* Finding the maximum depth for Decision Tree
* Finding the maximum depth for Random Forest

We find that the best model in terms of training convergence as well as testing accuracy is the **Random Forest** (even though SVM had a higher testing accuracy, it had less training accuracy, which sets it behind).

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

A picture containing text

Description automatically generated

A picture containing table

Description automatically generated

Table

Description automatically generated

With the Random Forest Model, we can also print its classification report.

Lastly, we use the Random Forest to predict the final testing dataset. In the **Predict and Save** section we basically call the test data and run our best model to get the output of our results which will be saved in the ‘leano\_result\_logistics3.csv’ for a final submission.

**Review Questions**

*Comparing the results, to that of a blind guess, explain why you think the results differed?*

The results would differ tremendously to a blind guess because creating a machine learning program allows it to be exposed to new data and independently adapt. Machine Learning is allowing the program to learn from previous computations to produce reliable, repeatable decisions and results. It would make no sense to collect all of this data and not be able to analyze and predict if a person will remain in the UDSU.

*Describe how you would improve your project if you had more time?*

If I had more time on my project, I definitely would have like to add another model, which would be Deep Neural Network. Based on what I have learned in this class, I would assume that Deep Neural Network versus SVM would have probably been my last two standing models to be determined as the best model.

I would have also like to look at more packages that could condense my program more. During the process of this project, I have learned about so many packages that have shortened my program by a lot. In the beginning my data cleaning had so many codes, but I realized throughout the time of writing it that certain parts were more important than others (ei. Looking at the shape of each variable, using the describe function) to get my point across. I definitely could have compacted my feature selection and modeling section, so I will be definitely looking into that in the future.