INST737: Milestone 3 Report

Team 8 – Sharvil Shastri, Sandy Staub, Erik Rye

Report
Explain additional data collection, data preparation and data cleaning efforts that you have done after Milestone 2. These additional efforts might include, for example, the collection of more tweets for a specific time range or the provision of labels for a given dataset of images.

Next, provide answers for the following five questions. If you believe that some of the requested methods are not applicable to your research question, please justify why.

Question 1. SVMs

In this question you are expected to explore how to apply SVMs to answer your research question(s).

a Train and test a linear SVM model

If your dependent variable is a class, you need to explore classification results for either binary or multiclass approaches, depending on your research question. For the multiclass, clarify the approach used (one-vs-one or one-vs-all).

If your dependent variable is a continuous feature, please frame your approach as a regression, not a classification. Please make sure you understand the nature of your problem before applying any random classifier.
b. Kernels

Replicate analyses in (a) for at least another type of non-linear kernel.

For both (a) and (b), report either the confusion matrix, the precision, recall, specificity, accuracy and the F1 measure for each class in your setting, or the RMSE for regression settings.

RQ1a: Our first research question is, "*Are we able to predict the number of applications a job posting could have based on the number of views it has received?*"
Our variables for this question are "Views (IV)" and "Applies (DV)". Since our dependent variable (DV) is a continuous feature, we must approach this question as a "Support Vector Regression (SVR)" instead of a "Support Vector Machine (SVM)" model. After removing records that were

N/A in one or both variables, we have 8069 records. We split the data into training and testing using an 80/20 split.
First, we trained the SVM Regression Model and produced predictions on the training data.

```
#Fit the SVM regression  model on the training data
svm_model <- svm(applies~views, data = jobpostings_Q1_train, kernel = "linear", epsilon = 0.1)
predictions_Q1_train <- predict(svm_model, jobpostings_Q1_train)
```

Then we evaluated the performance of the model using the training data.

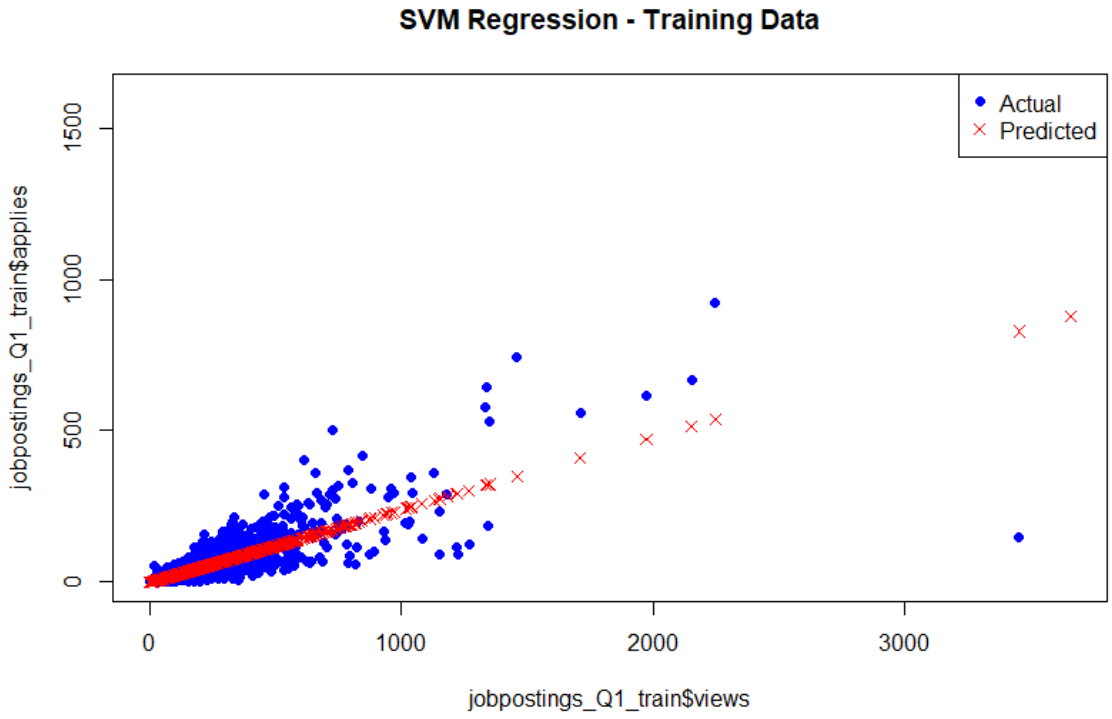
```
# Evaluate the performance (Mean Squared Error) of the training data
mse <- mean((jobpostings_Q1_train$applies - predictions_Q1_train)^2)
cat("Mean Squared Error on Training Data:", mse, "\n")
```

The Mean Squared Error on Training Data is 578.8266.

Next we plotted the training data and predictions:

```
# Plotting actual vs. predicted values for training data
predicted_applies_train <- predict(svm_model, jobpostings_Q1_train)
plot(jobpostings_Q1_train$views, jobpostings_Q1_train$applies, col = "blue", pch = 16,
     main = "SVM Regression - Training Data")
points(jobpostings_Q1_train$views, predicted_applies_train, col="red", pch=4)
legend("topright", legend = c("Actual", "Predicted"), col = c("blue", "red"), pch = c(16, 4))
```

In the graph below, the Training data “views” vs “applies” actual values are represented in blue. The predicted values are represented in red.

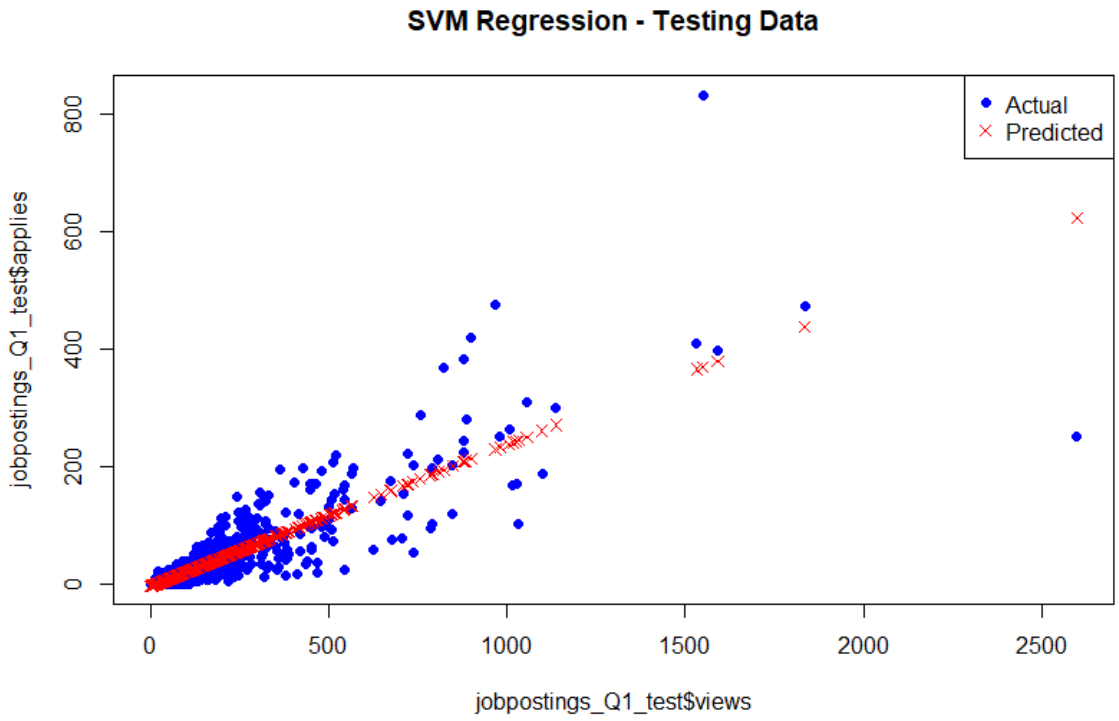


For the testing data, we run the same steps:

```r
# Make predictions on the testing data
predictions_Q1 <- predict(svm_model, jobpostings_Q1_test)

# Evaluate the performance (Mean Squared Error) of the testing data
mse <- mean((jobpostings_Q1_test$applies - predictions_Q1)^2)
cat("Mean Squared Error on Testing Data:", mse, "\n")
```

```r
# Plotting actual vs. predicted values for testing data
predicted_applies_test <- predict(svm_model, jobpostings_Q1_test)
plot(jobpostings_Q1_test$views, jobpostings_Q1_test$applies, col = "blue", pch = 16,
     main = "SVM Regression - Testing Data")
points(jobpostings_Q1_test$views, predicted_applies_test, col = "red", pch = 4)
legend("topright", legend = c("Actual", "Predicted"), col = c("blue", "red"), pch = c(16, 4))
```

The following similar graph was generated. The Mean Squared Error on Testing Data is 634.2471. This would imply that the training data performed better than the testing data.



RQ1b:

We replicated our analyses for our first question, using the non-linear kernel of “Radial Basis Function (RBF)”

The first step is to create the regression model with an RBF kernel, using the existing data frame, training data and testing data. Then we made predictions for both the training and testing data sets.

```
#Using the same data frame and the same training and testing data sets
# Create SVM regression model with an RBF kernel
svm_model_rbf <- ksvm(applies ~ ., data = jobpostings_Q1_train, type = "eps-svr", kernel = "rbfdot")

# Make predictions on the training and testing data
predictions_train_Q1 <- predict(svm_model_rbf, jobpostings_Q1_train)
predictions_test_Q1 <- predict(svm_model_rbf, jobpostings_Q1_test)
```

Then we evaluate the model on both training sets using MSE.

```
# Evaluate the model performance on training data using MSE
mse_train_Q1 <- mean((jobpostings_Q1_train$applies - predictions_train_Q1)^2)
cat("MSE on Training Data (RBF Kernel):", mse_train_Q1, "\n")

# Evaluate the model performance on testing data using MSE
mse_test_Q1 <- mean((jobpostings_Q1_test$applies - predictions_test_Q1)^2)
cat("MSE on Testing Data (RBF Kernel):", mse_test_Q1, "\n")
```

MSE on Training Data (RBF Kernel): 922.5892
MSE on Testing Data (RBF Kernel): 1114.752
These results are much worse than the linear kernel, and still the training data did better than the testing data. To compare, we then converted the MSE to RMSE and returned the following values:
RMSE on Training Data (RBF Kernel): 33.38791
RMSE on Testing Data (RBF Kernel): 30.37415

RQ2a:Our second research question is, *"Can we predict the number of applicants based on the job's salary information?"*
In our original dataset, both the number of applicants and the salary data were continuous. For milestone 2, we created classes for these variables. In order to try a multiclass approach, we reverted to the class_applies for the dependent variable in this question, and added in additional variables of . "Class_applies" has values of Low, Medium, High, and Unspecified. The Independent variable is "clean_salary". We added "formatted_work_type" and "formatted_experience_level" for the multiclass approach.
First we create a new data frame for the fields that we are working with.

```
#QUESTION 2
#create dataframe for Question 2 for only the variables we are using, to simplify
jobpostings_Q2 <- data.frame(clean_salary,class_applies,formatted_work_type, formatted_experience_level)
View(jobpostings_Q2)
```

Then we convert "clean_salary" to a factor so we can use it in the multiclass approach.
Once again we remove NAs from the data so that the arguments all have the same length.

```
#change clean_salary to a factor
jobpostings_Q2$clean_salary <- as.factor(jobpostings_Q2$clean_salary)

#drop na's so the arguments have the same length
jobpostings_Q2 <- drop_na(jobpostings_Q2)
#Just checking to make sure it worked
View(jobpostings_Q2)
```

Then we created the classifier to train the model:

```
#create classifier to train the SVM model
applies_classifier <-ksvm(clean_salary~.,data=jobpostings_Q2_train, kernel="vanilladot", type="C-svc", C=1)
applies_classifier
```

When we attempted to used the independent variables of “clean_salary”, “formatted_work_type” and “formatted_experience_level” for the multiclass approach, we ended up with a training error of 1:

```
Support Vector Machine object of class "ksvm"
SV type: C-svc  (classification)
parameter : cost C = 1
Linear (vanilla) kernel function
Number of Support Vectors : 531
Training error : 1
```

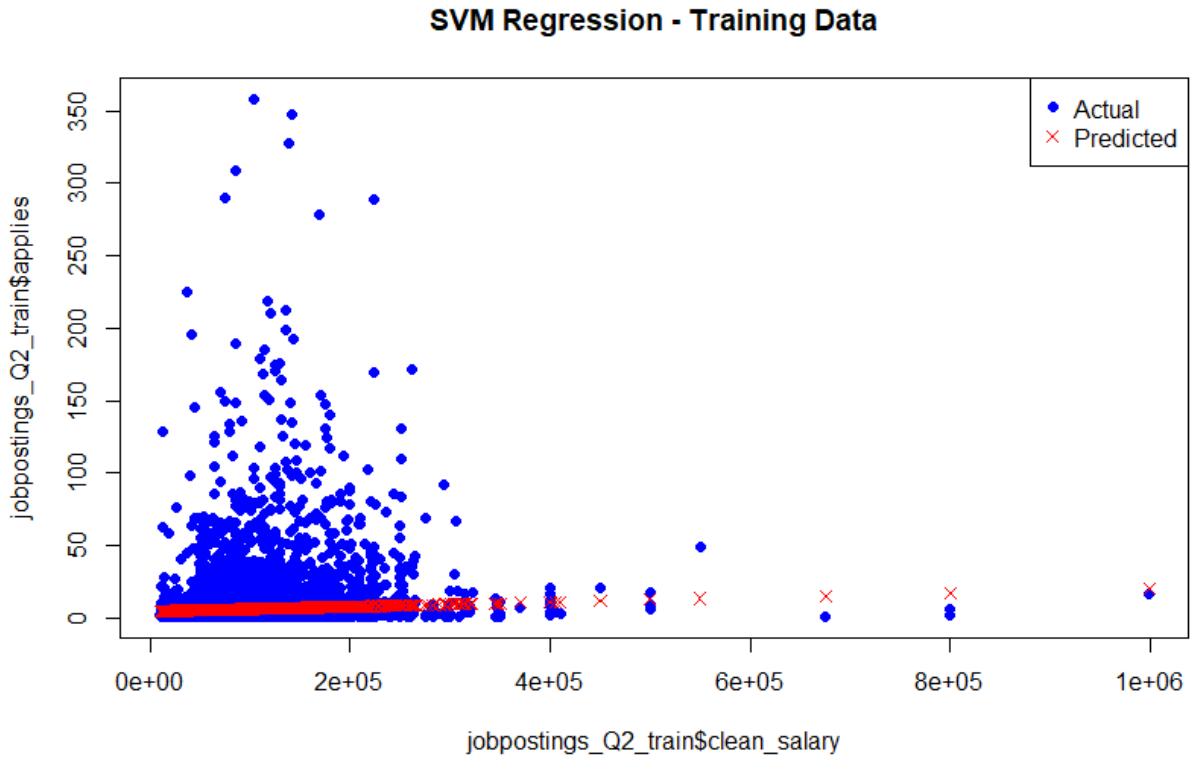
This implied that the SVM model achieved perfect accuracy on the training data. A training error of 1 means that every data point in the training set was correctly classified by the model. This looked like a sign of potential overfitting. In addition, the fact that the number of Support Vectors was almost identical to the number of records in the dataset, was a sign that the model was memorizing the training data rather than being generalized for use in testing data.

So, instead we went back to the continuous variable for “applies” and did an SVM regression model of “clean_salary” and “applies”. We created the specific data frame for this question, dropped the NAs and separated the data into training and testing on an 80/20 split of those that were remaining after the NAs were removed.

```
#####QUESTION 2########
#create dataframe for Question 2 for only the variables we are using, to simplify
jobpostings_Q2 <- data.frame(clean_salary,applies)
View(jobpostings_Q2)

#drop na's so the arguments have the same length
jobpostings_Q2 <- drop_na(jobpostings_Q2)
#Just checking to make sure it worked
View(jobpostings_Q2)
#separate the data into training and testing sets (80/20)
jobpostings_Q2_train <- jobpostings_Q2[1:2653,]
jobpostings_Q2_test <- jobpostings_Q2[2654:3316,]
View(jobpostings_Q2_train)
View(jobpostings_Q2_test)
```

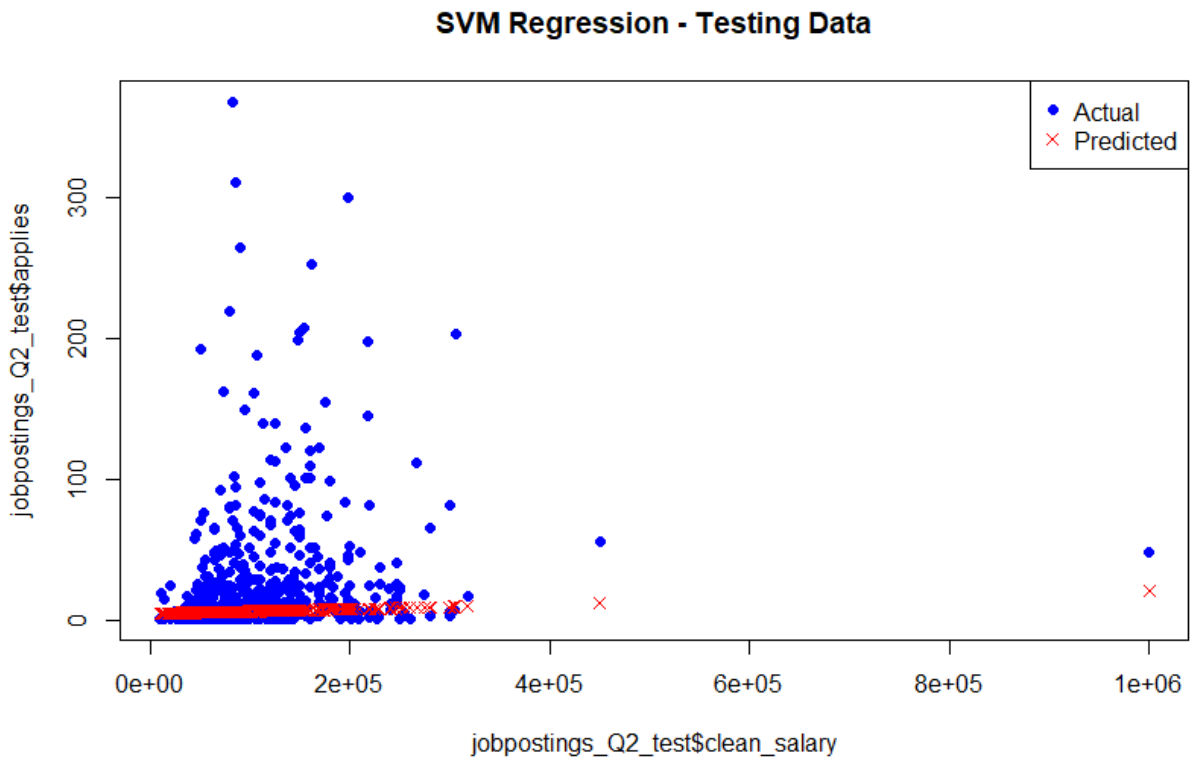
Next, we fit the SVM regression model onto the training data and run the related prediction. After that, we evaluate the performance of the model using Mean Square Error (MSE) on the training data. Then we repeat the evaluation and prediction on the testing data.
Afterwards, we compare the MSE values of the training data and testing data , which were 1027.15 and 1978.252, respectively. Once again, this metric was better for the training data than the testing data.

```r
#Fit the SVM regression  model on the training data and run prediction
svm_model_Q2 <- svm(applies~clean_salary, data = jobpostings_Q2_train, kernel = "linear", epsilon = 0.1)
predictions_Q2_train <- predict(svm_model_Q2, jobpostings_Q2_train)

# Evaluate the performance (Mean Squared Error) of the training data
mse_Q2 <- mean((jobpostings_Q2_train$applies - predictions_Q2_train)^2)
cat("Mean Squared Error on Training Data:", mse_Q2, "\n")

# Make predictions on the testing data
predictions_Q2 <- predict(svm_model_Q2, jobpostings_Q2_test)

# Evaluate the performance (Mean Squared Error) of the testing data
mse_Q2_test <- mean((jobpostings_Q2_test$applies - predictions_Q2)^2)
cat("Mean Squared Error on Testing Data:", mse_Q2_test, "\n")
```



Mean Squared Error on Training Data: 1027.15

```r
# Plotting actual vs. predicted values for testing data
predicted_applies_test_Q2 <- predict(svm_model_Q2, jobpostings_Q2_test)
plot(jobpostings_Q2_test$clean_salary, jobpostings_Q2_test$applies, col = "blue", pch = 16,
     main = "SVM Regression - Testing Data")
points(jobpostings_Q2_test$clean_salary, predicted_applies_test_Q2, col = "red", pch = 4)
legend("topright", legend = c("Actual", "Predicted"), col = c("blue", "red"), pch = c(16, 4))
```



Mean Squared Error on Testing Data: 1978.252

In the graphs above, the salary has been represented by E notation. For example, 2e+05 = $200,000.
Based on the MSE values and the plots above, there appears to be no relationship between the salary and the number of applicants.

Question 2. Neural Networks
In this question, you will explore the applicability of Neural Networks to your research question(s). Remember to normalize the input variables to [0-1] ranges.

You are expected to do an exhaustive evaluation of the NN prediction results for different hidden layers and report: (1) performance results, (2) activation function and (3) number of neurons per layer for each case. Also, plot the best neural network model.

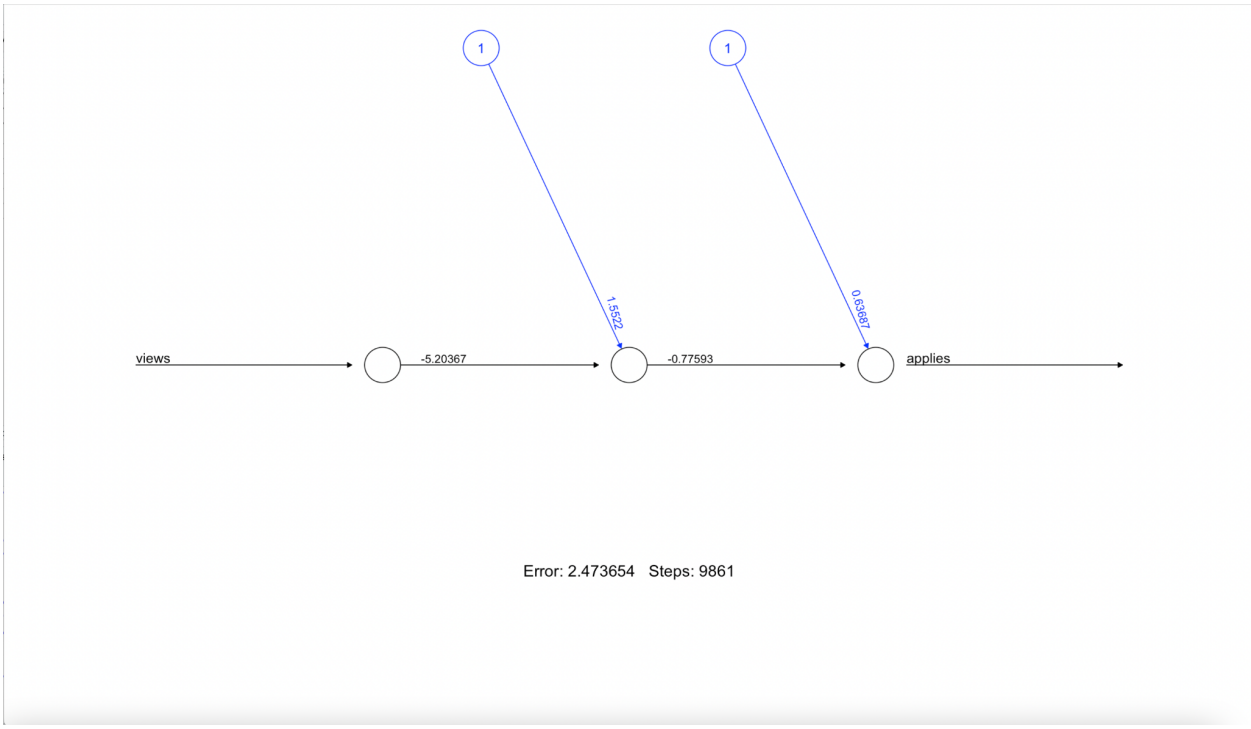
For Milestone 2, we had to categorize some of our continuous variables, so that we can use Logistic Regression & Decision Trees on them. But since Neural Networks can work with continuous & quantitative variables, we decided to forego those newly created categories and only work with the original continuous variables.

For one of our original research questions, which was to predict the number of applicants based on the number of applicants a job posting will have, based on the number of views it has.

The data was first normalized, and it looked like this:

	views	salary	employees	followers	applies
1	0.0008183306	0.080606061	0.064520915	0.064614554	0.000000000
2	0.0073649755	0.082343434	0.064520915	0.064614554	0.002724796
3	0.0163666121	0.399595960	0.085627859	0.114351788	0.008174387
4	0.0000000000	0.195050505	0.120221538	0.193893336	0.000000000
5	0.0196399345	0.217575758	0.120221538	0.193893336	0.008174387
6	0.5744680851	0.259494949	0.120221538	0.193893336	0.302452316
7	0.1014729951	0.315151515	0.120221538	0.193893336	0.046321526

The corresponding Neural Network model, with the default attributes provided a correlation coefficient of around 0.89. This is the plot.



After several iterations, playing around with different activation functions, hidden layers, number of nodes in each, number of maximum steps and total number of variables, the best Neural Network model was achieved by using the following parameters: Threshold activation function (default), 4 hidden layers, 3 nodes in the first & second layers, 2 nodes in the third & fourth

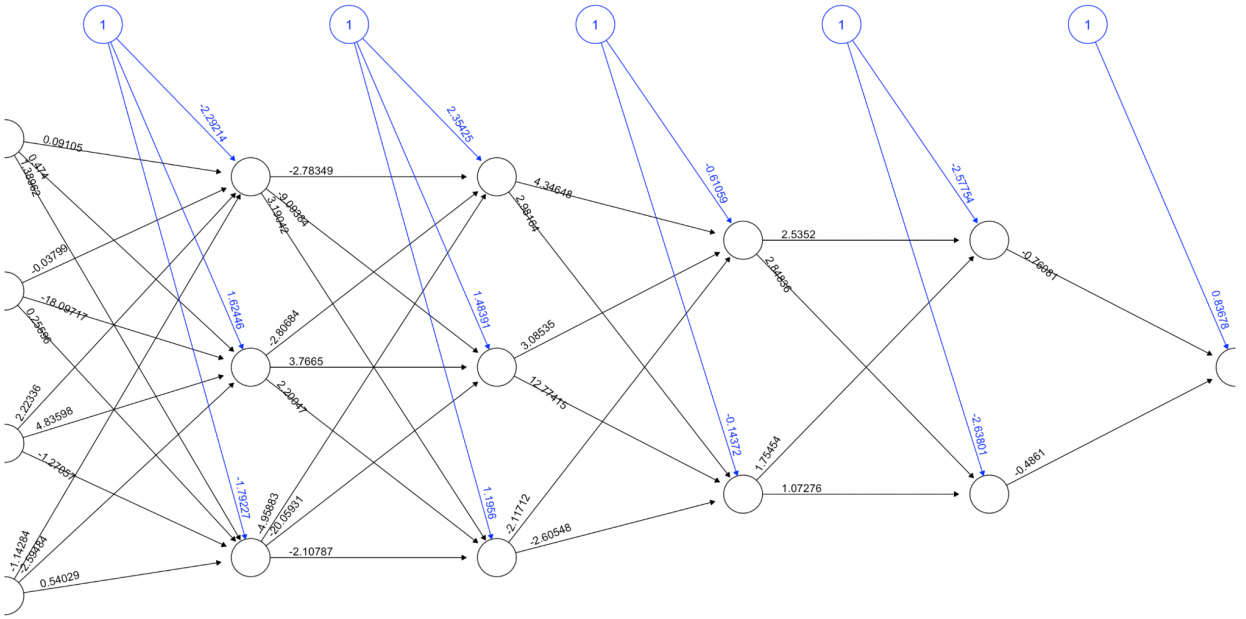
layers, and using continuous variables - the number of views, salary, employees working at the company & the respective followers.
Even with using sigmoid and ReLU activation, the model performance hardly improved.

```r
# 4 hidden layers, 3 nodes in first & second layer, 2 nodes in third & fourth Layer
set.seed(123)
applicants_model_main <- neuralnet(formula = applies ~ views + salary + employees + followers, data = applicants_train,
hidden = c(3,3,2,2))

#Plotting the Neural Network Model
plot(applicants_model_main)
model_main_results <- neuralnet::compute(applicants_model_main, applicants_test[1:4])

predicted_applicants_model_main <- model_main_results$net.result
cor(predicted_applicants_model_main,applicants_test$applies)
```

```
          [,1]
[1,] 0.9128009
```

The performance of the Neural Network model was improved slightly. The value of correlation coefficient came out around 0.9128009.

The plot of the best Neural Network model:

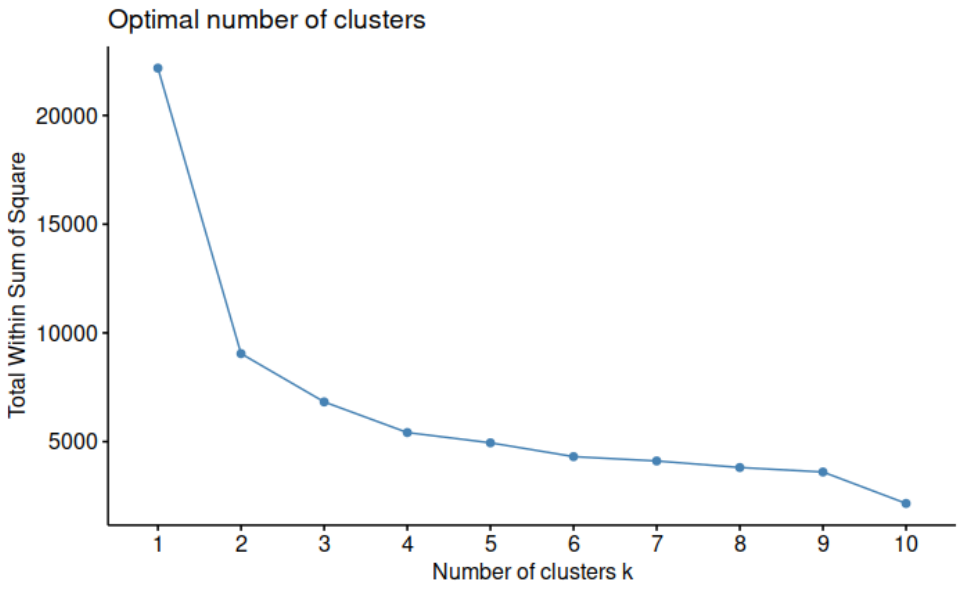


Question 3. Clustering

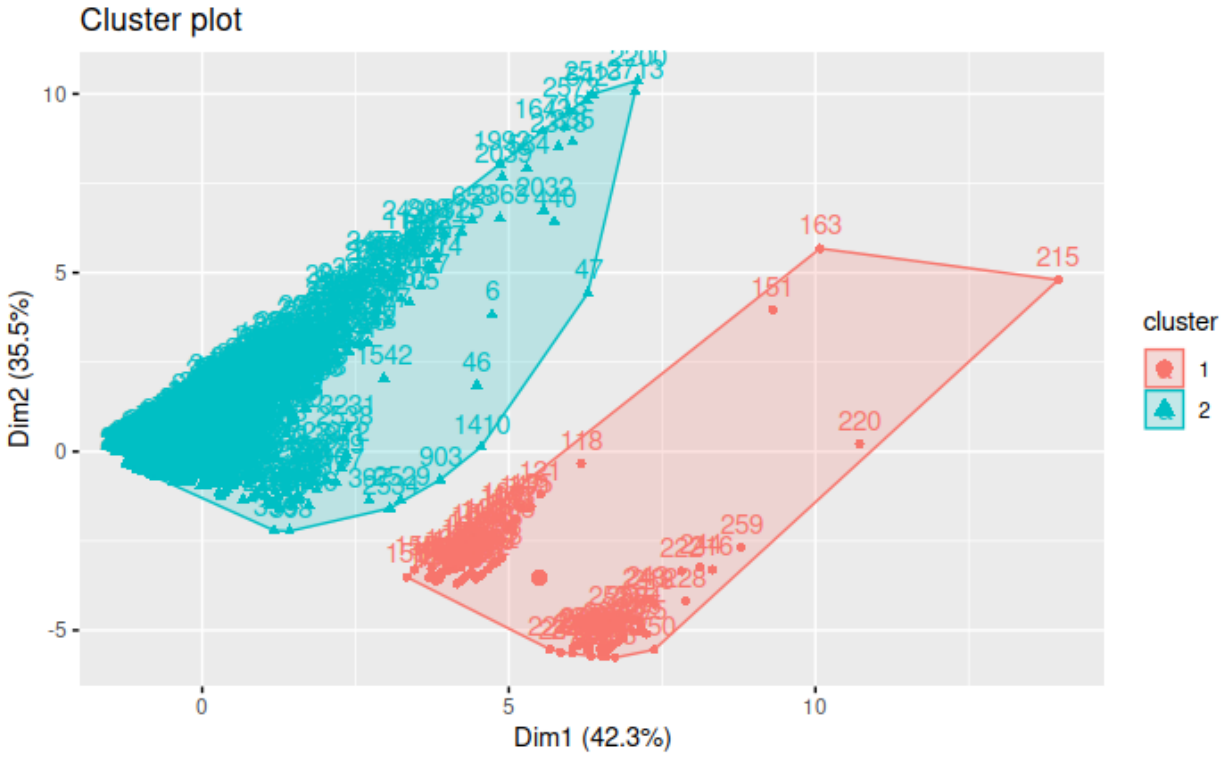
For each of the clustering techniques, we used the number of job listing views, applicants, normalized salary, number of company employees, and number of company LinkedIn followers. We normalized these numerical values using the `scale()` function before applying the clustering techniques. We describe each of the three clustering methods we used in the following subsections:

1. **K-Means Clustering**

The first clustering technique we investigated was k-means clustering. For k-means clustering, we found that the best number of clusters was 2, which we determined by looking at the plot produced by `fviz_nbclusters()`. After 2 clusters, the reduction in distance between the points within the cluster decreases at a decreasing rate, indicating that there are diminishing returns in using additional clusters. In common parlance, 2 is where the “knee” in the plot appears; with the “knee” indicating the optimal number of clusters.



When using two clusters, the k-means clustering algorithm's best run produced one cluster with 126 elements (red in the following plot), and another with 3,125 elements (blue in the below plot).

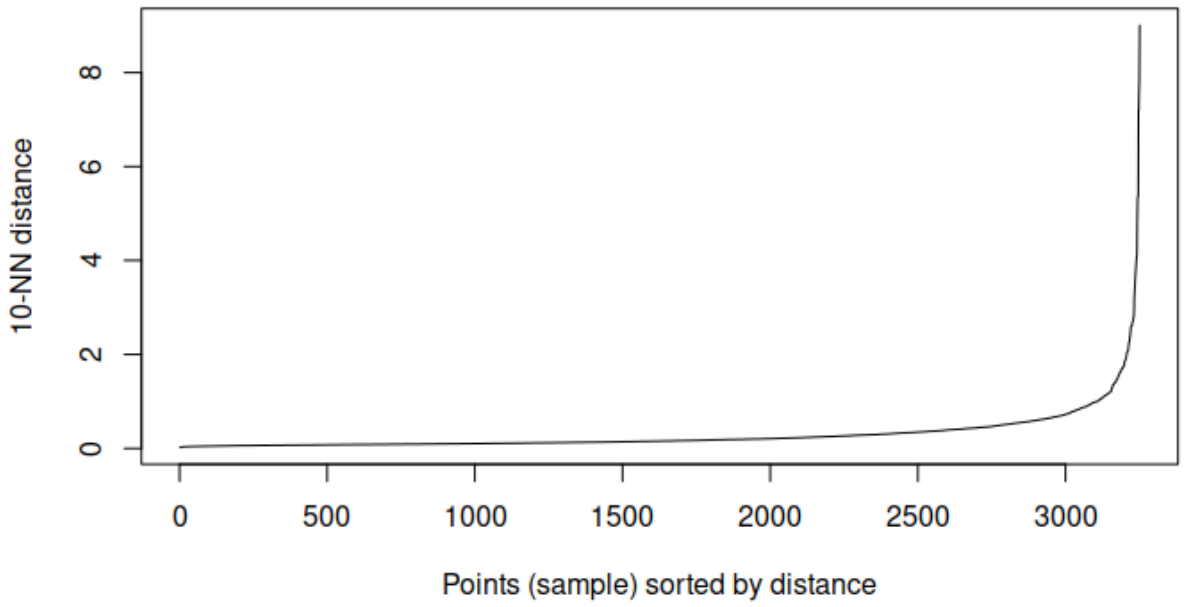


In order to interpret the meaning of the clusters, we examined the mean feature values per cluster by looking at the cluster centers using the `centers` column of the kmeans cluster dataframe. The following table displays the mean (normalized) value for each of our dependent variables. The larger cluster (1) has significantly higher views, salary, employees, and followers than does cluster 2.

	views	salary	applies	employees	followers
1	0.5220470	1.39323449	0.008183608	6.0570426	8.18618755
2	0.1036605	0.08016685	-0.043341035	-0.1269067	-0.07660683

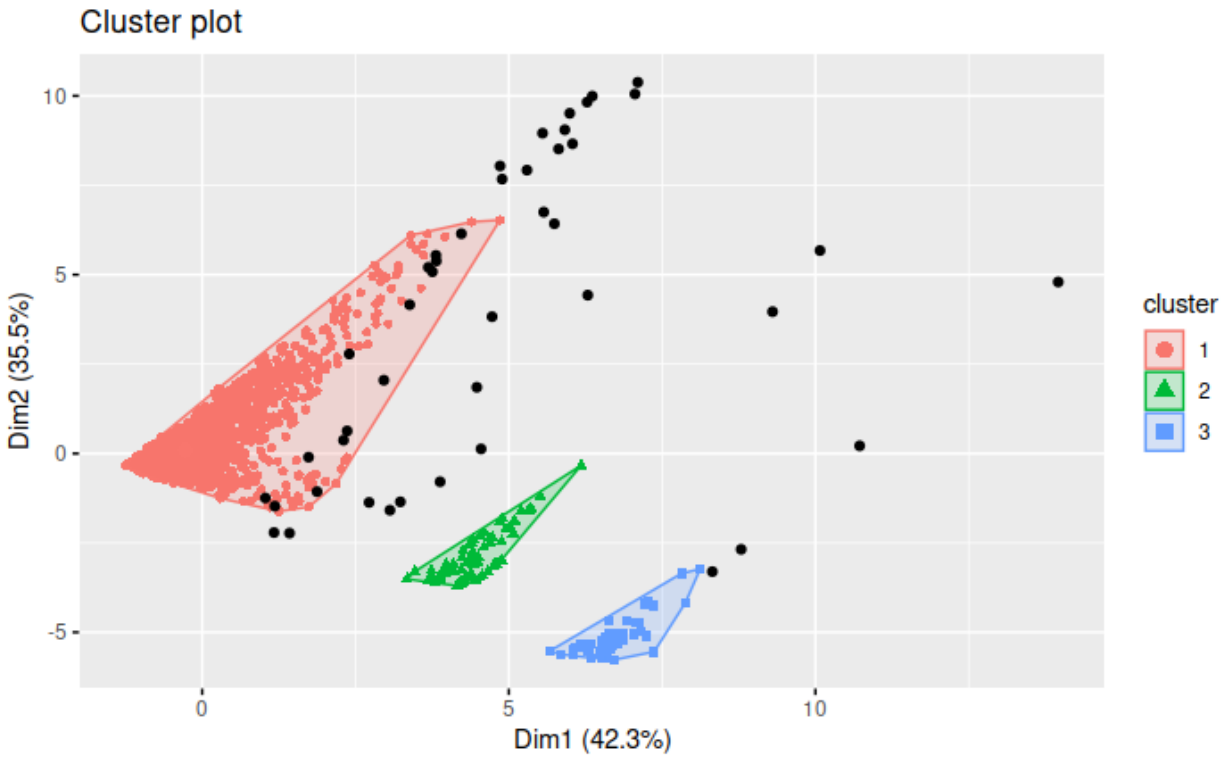
2. Density-Based Clustering (DBSCAN)

Next, we looked at density-based clustering methods. We arbitrarily selected 10 as the minimum number of jobs in each cluster we would like to see, and then used the `dbscan` package's `kNNdistplot` method to determine the correct epsilon value to use for DBSCAN's clustering algorithm.



The plot that this produced sorts the 10-clusters by distance; the optimal epsilon value for the specified number of nearest neighbors in each cluster appears where there is a knee in the plot. This occurs at a distance of about 1.5, which we used as our epsilon value for the clustering algorithm.

Using 10 as the number of nearest neighbors required for a cluster generates three distinct clusters.



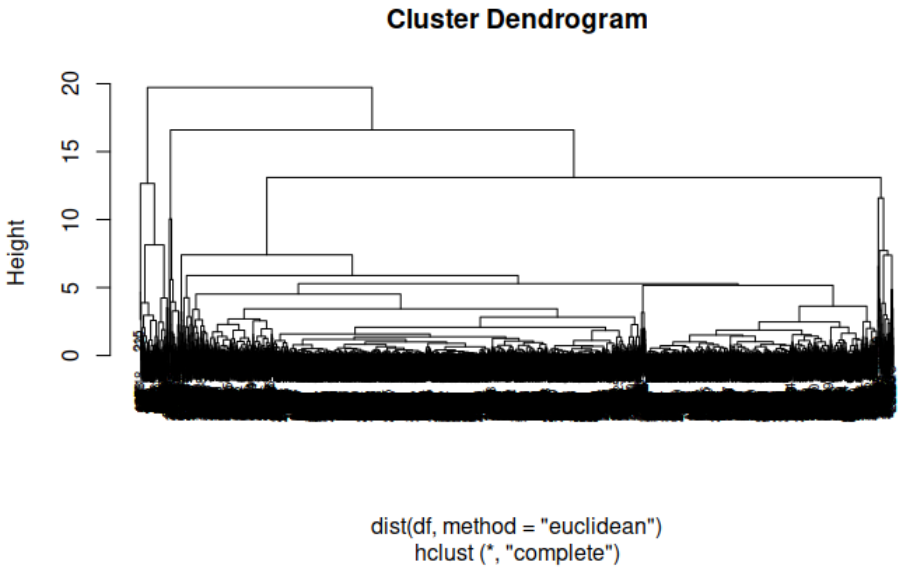
Using k=10, there are 44 noise points, which are contained in no cluster and depicted as black dots in the preceding figure. Cluster 1, the red cluster, is the largest cluster, containing 3,087 distinct points. Cluster 2 and 3 contain 75 and 45 points each, respectively.

We then manually computed the mean variable values per cluster. The largest cluster, cluster 1, has normalized values near zero for all five of the dependent variables we considered. Cluster 2 contains points with markedly higher values across all variables except the number of applicants, `applies`. Cluster 3 contains points with a higher number of views and salary than group 1, but lower than group 2. Group 3's mean number of employees is far higher than either cluster 1 or 2.

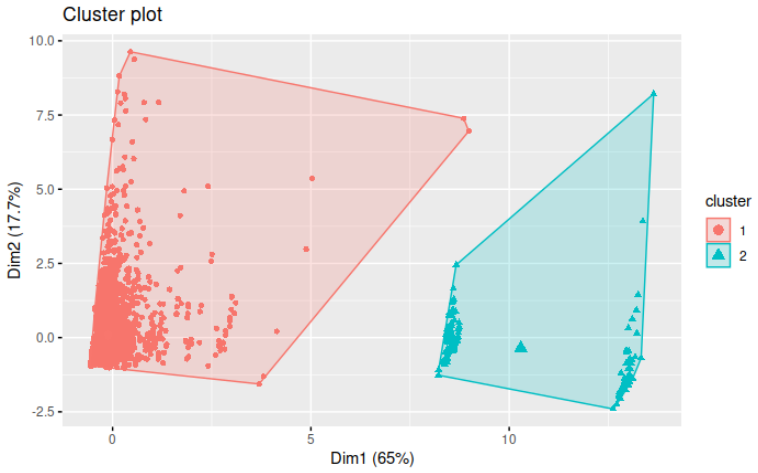
Group.1	views	salary	applies	employees	followers
1	0.0582797	0.05586911	-0.07496202	-0.1358995	-0.08440245
2	0.3731186	1.67756375	-0.14379705	3.3704604	8.06843287
3	0.1281671	0.91491140	-0.13612666	10.2788147	8.37123060

3. Hierarchical Clustering

Finally, we used hierarchical clustering to cluster our data. As with the previous two methods, we first normalized our five continuous variables. Then, we created a dendrogram using the Euclidean distance metric, as shown in the plot below. Our dendrogram looks chaotic and is not very useful due to the number of rows in our data — 3,251 when all rows with NA values are removed across all five dependent variables.



When we trim the dendrogram at k=2 to produce two clusters, however, two clear clusters appear in the resulting cluster plot. The main cluster, Cluster 1, is depicted in red in the following plots. It is significantly larger than Cluster 2, with 3,127 points as compared with Cluster 2's 124 points. The two clusters have similar means for the number of applications the job posting received; cluster 2, however, had much higher means for the "views", "salary", "employees" and "followers" variables.

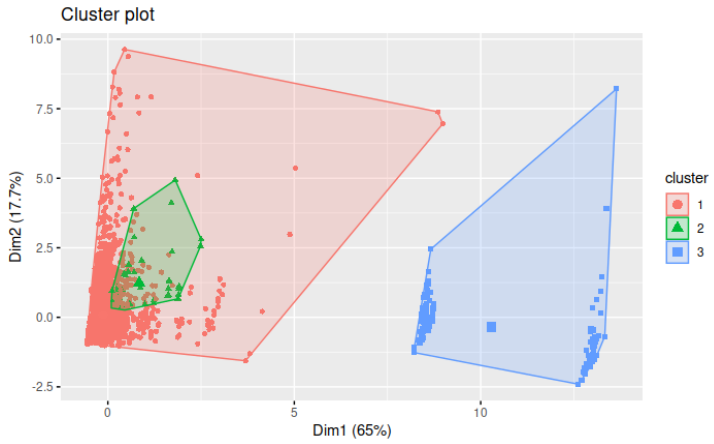


Description: df [2 × 6]

cluster <int>	views <dbl>	salary <dbl>	applies <dbl>	employees <dbl>	followers <dbl>
1	0.1083219	0.08077214	-0.04092380	-0.1246698	-0.07139734
2	0.4112465	1.39914896	-0.05194249	6.1003746	8.18808681

2 rows

When we trim the dendrogram into 3 or more clusters, we begin to see clusters appear entirely contained within another. This indicates that there is either a strongly related subset of one cluster, or that there is limited utility in continuing to subdivide the data into further clusters. Based on the visualization below, we believe that the data shows that it is likely the second case, as the green cluster within the red cluster does not appear to be more tightly related than the surrounding red points.



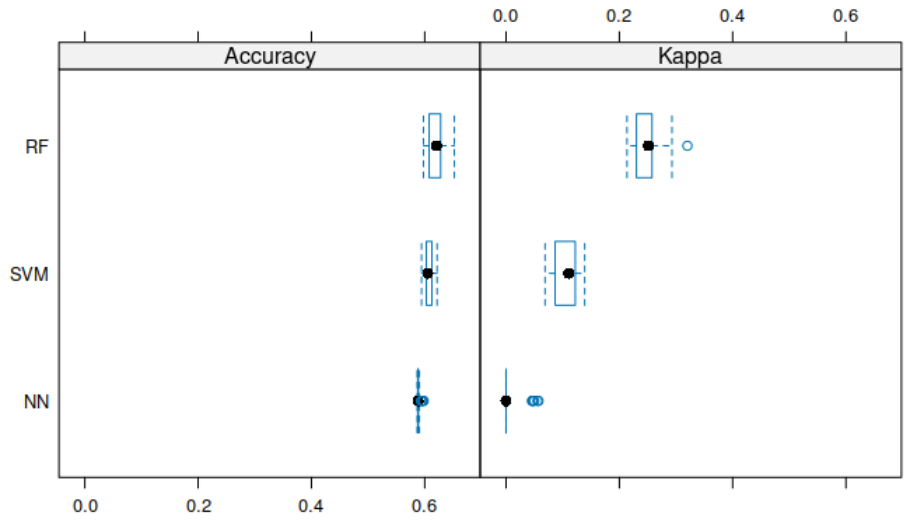
Question 4. Comparative Analysis

Take one of your research questions focusing on a classification task (dependent variables are classes). If all your questions have a continuous dependent variable, create classes by dividing it into ranges. Next, using the CARET package, compare the performance of the models from Question 1 and 2 in this Milestone and Question 3 in Milestone 2. You are expected to compare results across classifiers but also across different cross-validation techniques as seen in class (use at least three different CV approaches). Discuss the results.

For this question, we chose to look at one of our classification tasks that attempts to predict the class of salary (“high”, “medium”, or “low”) based on the independent variables of number of views a job posting received, the number of applicants it generated, the number of employees the company has, and the number of LinkedIn followers the company has.

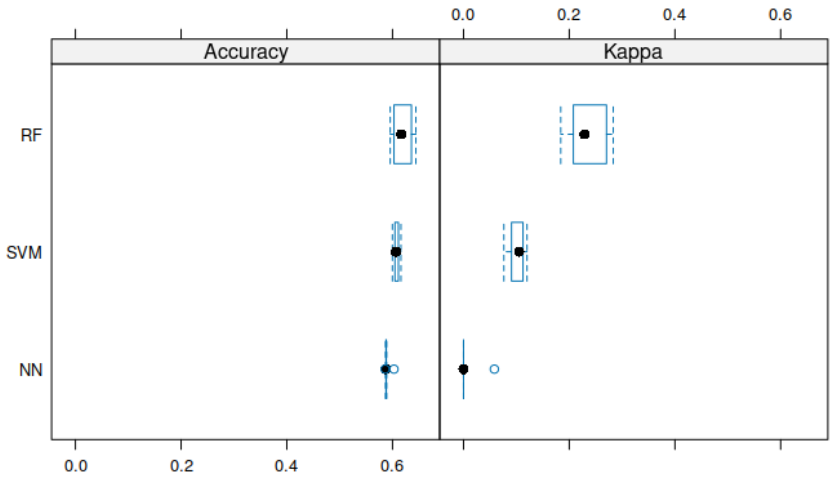
For our first test, we compared the performance of a Random Forest, SVM, and Neural Network

classifier using a single 10-fold cross validation. The Random Forest model performs best of the three models we compare, with an accuracy of 0.62. The SVM model performs similarly, with an accuracy of 0.61. The Neural Network classifier performs the worst, with a median accuracy of 0.58. Examining the Kappa's constant for the three models indicates that both the SVM and Neural Network models' accuracy is due primarily to random chance, as the median Kappa constants for the SVM mode is about 0.1 and the Neural Network's is 0. The Random Forest model, however, has a much higher median Kappa's constant (0.27), indicating that there is much better agreement among the Random Forest model output.




```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 30

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.5868188 0.5875635 0.5881981 0.5891053 0.5887019 0.5982256    0
RF  0.5977157 0.6085025 0.6209526 0.6196836 0.6271015 0.6522843    0
SVM 0.5944233 0.6027919 0.6053293 0.6073323 0.6123418 0.6218274    0

Kappa
          Min.    1st Qu.    Median       Mean   3rd Qu.       Max. NA's
NN  0.00000000 0.00000000 0.0000000 0.01007709 0.0000000 0.05722724    0
RF  0.21353036 0.23113012 0.2509991 0.25051556 0.2576615 0.32013993    0
SVM 0.06881367 0.08704725 0.1103868 0.10375520 0.1209983 0.13888940    0
```

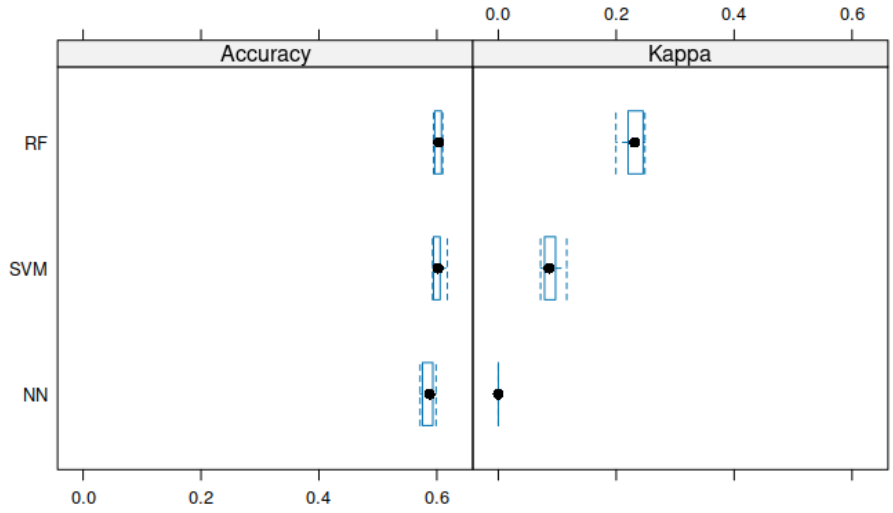
Next, we repeated the experiment using the “repeatedcv”, or repeated cross validation technique. We again chose 10 folds, but repeated the cross-validation three times. The performance of the three models is roughly the same, with the Neural Network classifier performing the worst, with a 0.59 median accuracy, and the Random Forest model performing best, with an 0.62 median accuracy. The Kappa’s constants for each indicate that both the SVM and Neural Network classifiers have relatively poor agreement between different models (both have median Kappa’s constants well below 0.2, with the Neural Network’s median Kappa again at 0), while the Random Forest model has a significantly higher median Kappa’s constant of about 0.23. This indicates that there is more agreement between Random Forest output than the other two models.




```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.5860759 0.5875635 0.5875635 0.5890646 0.5881234 0.6027919    0
RF  0.5956907 0.6043782 0.6163620 0.6181207 0.6316624 0.6442186    0
SVM 0.5997459 0.6047545 0.6064639 0.6070771 0.6107944 0.6159696    0

Kappa
          Min.    1st Qu.    Median       Mean   3rd Qu.      Max. NA's
NN  0.00000000 0.00000000 0.0000000 0.00583627 0.0000000 0.0583627    0
RF  0.18391856 0.20954044 0.2288476 0.23392712 0.2614739 0.2832845    0
SVM 0.07634959 0.09247061 0.1050776 0.10216150 0.1118015 0.1201178    0
```

The results of our repeated cross-validation roughly match those of our single cross validation, with the median accuracy and Kappa's coefficient decreasing very slightly for each of the models.

Finally, we used the bootstrap method after attempting to use the LOOCV, which was too computationally intensive for the machine we ran the model on. The bootstrap method draws samples from the dataset with replacement to generate samples; the values not sampled become the test data. Again, the performance order of our three models is Random Forest, SVM, and Neural Network , with Random Forest and SVM performing about as well as each other. However, Random Forest has a much higher Kappa statistic than either SVM or Neural Network, which indicates that its accuracy is much less due to random chance than the other two models.




```
Models: NN, RF, SVM
Number of resamples: 10

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.5716268 0.5777215 0.5877074 0.5859621 0.5925799 0.5986727    0
RF  0.5943983 0.5977125 0.6032575 0.6026612 0.6075341 0.6100136    0
SVM 0.5927186 0.5952212 0.6022327 0.6021181 0.6051650 0.6178101    0

Kappa
        Min.    1st Qu.     Median       Mean    3rd Qu.      Max. NA's
NN  0.000000 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000    0
RF  0.199485 0.22194764 0.23115423 0.23066790 0.24417280 0.2487441    0
SVM 0.071666 0.07921603 0.08572228 0.08826025 0.09546908 0.1160016    0
```

In general, between all three validation methods, we found that there was a significant amount of agreement between the accuracies of the classifiers as well as their Kappa coefficients.

Question 5. Feature Selection

Select three models from Milestones 2 or 3 (linear/logistic regression, Naïve Bayes, Decision Trees, SVM or Neural Networks) and apply three types of feature selection: filter, wrapper or embedding (when applicable). Report results focusing on improvements in performance due to feature selection.

For Feature Selection, we would be considering the following three models:

- Multivariate Regression
- Neural Networks
- Decision Trees

a
1. Wrapper SFS and SBS for Multivariate Regression:
For Milestone 2, in the Multivariate Regression model, the Dependent variable was the number of applicants & the independent variables were the number of views, salary, the experience level, the work type & the state.

The summary of the model was as follows:

```
state AZ Area                              2.420760475 23.488036747   0.103            0.917922
state CA                                  -4.325865100 16.620681747  -0.260            0.794679
state CO                                  -2.431122954 16.708807410  -0.145            0.884329
state CT                                   5.351025225 16.966165873   0.315            0.752492
state DC                                  -1.825606236 16.922463041  -0.108            0.914100
state DE                                   2.309602123 23.461056712   0.098            0.921588
state FL                                   1.011645133 16.696487900   0.061            0.951691
state GA                                  -0.934569496 16.728737115  -0.056            0.955453
state HI                                  -2.048311992 20.343498631  -0.101            0.919808
state IA                                  -5.164873258 17.520029322  -0.295            0.768175
state ID                                  -3.605466653 17.601593910  -0.205            0.837717
state IL                                  -1.899515525 16.704909834  -0.114            0.909478
state IN                                   2.047298764 17.117750433   0.120            0.904810
state KS                                  -3.539298712 18.559748305  -0.191            0.848780
state KY                                   0.039444371 17.749704776   0.002            0.998227
state LA                                  -3.598520736 17.614851594  -0.204            0.838146
state MA                                  -3.081000134 16.748298046  -0.184            0.854062
state Massachusetts Metropolitan Area     -8.125410242 20.368818029  -0.399            0.689994
state MD                                  -1.875069316 16.839024997  -0.111            0.911347
state ME                                   6.541849424 20.345670332   0.322            0.747835
state MI                                  -2.753651984 16.842919058  -0.163            0.870147
state MN                                  -5.105270182 16.909644215  -0.302            0.762745
state MO                                  -5.665179527 17.066410806  -0.332            0.739958
state MS                                  -1.959324797 23.478014186  -0.083            0.933498
state MT                                  -5.830095492 20.328672398  -0.287            0.774298
state NC                                  -0.115309278 16.774465919  -0.007            0.994516
```

```
state NY                                -4.964499181 16.649006956  -0.298             0.765389
state OH                                -1.024526872 16.814213600  -0.061             0.951419
state Ohio Metropolitan Area            -1.729632433 18.564730581  -0.093             0.925779
state OK                                -1.825767504 18.554767744  -0.098             0.921624
state OR                                -3.637538953 17.117775496  -0.213             0.831736
state Oregon Metropolitan Area          37.467109130 18.214155269   2.057             0.039797 *
state PA                                 2.539356380 16.783518953   0.151             0.879752
state RI                                -1.641683422 17.604896192  -0.093             0.925712
state SC                                -2.984070869 16.928939515  -0.176             0.860097
state South Carolina Area               -5.148915982 23.495176596  -0.219             0.826555
state South Carolina Metropolitan Area  -3.067100493 18.594240407  -0.165             0.868999
state Texas Metropolitan Area            1.188793234 17.090047920   0.070             0.944550
state TN                                -5.085953432 16.880844870  -0.301             0.763224
state TX                                -1.037655560 16.659397628  -0.062             0.950340
state United States                     -2.952731479 16.666646201  -0.177             0.859395
state UT                                -0.686983163 16.957550871  -0.041             0.967689
state VA                                -2.319607625 16.733365306  -0.139             0.889761
state WA                                -3.837292231 16.688551234  -0.230             0.818163
state WI                                 1.597683808 16.898442950   0.095             0.924684
state WV                                -0.535214407 23.460340024  -0.023             0.981801
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.59 on 2265 degrees of freedom
Multiple R-squared:  0.7662,    Adjusted R-squared:  0.7592
F-statistic: 109.2 on 68 and 2265 DF,  p-value: < 0.00000000000000022
```

(The independent variables contain both categorical & continuous variables)
The R-squared value for the multivariate regression model was around 0.7592.

Wrapper - Sequential Forward & Backward Selection (**SFS & SBS**) Feature Selection was carried out for the Multivariate Regression Model, using step() function.
step() function in R searches for the best possible model by iteratively selecting features to arrive at a model with the lowest possible AIC (Akike Information Criterion)

The shortlisted variables for **both** cases were:

```
 [1] "views"                                  "formatted_experience_levelDirector"
 [3] "formatted_experience_levelEntry level"  "formatted_experience_levelExecutive"
 [5] "formatted_experience_levelInternship"   "formatted_experience_levelMid-Senior level"
 [7] "formatted_work_typeFull-time"           "formatted_work_typeInternship"
 [9] "formatted_work_typeOther"               "formatted_work_typePart-time"
[11] "formatted_work_typeTemporary"           "clean_salary"
```

The summary of the Multivariate Regression Model, with these corresponding shortlisted variables is:

```
Call:
lm(formula = applies ~ views + formatted_experience_level + formatted_work_type +
    clean_salary, data = dataset_FS1)

Residuals:
     Min       1Q   Median       3Q      Max
-204.365   -4.742    0.011    3.124  190.156

Coefficients:
                                          Estimate  Std. Error t value             Pr(>|t|)
(Intercept)                             3.67644429  1.24096945   2.963              0.00308 **
views                                   0.25138427  0.00302523  83.096 < 0.0000000000000002 ***
formatted_experience_levelDirector     -5.14992583  1.61785771  -3.183              0.00148 **
formatted_experience_levelEntry level  -0.34954847  1.11534193  -0.313              0.75400
formatted_experience_levelExecutive    -2.48760503  3.13309823  -0.794              0.42729
formatted_experience_levelInternship   -4.23773415  3.24000096  -1.308              0.19102
formatted_experience_levelMid-Senior level  0.87618149  0.98333837   0.891              0.37301
formatted_work_typeFull-time           -4.32406103  0.99465940  -4.347            0.0000144 ***
formatted_work_typeInternship          -5.31120205  6.40214695  -0.830              0.40685
formatted_work_typeOther               -7.76973633  5.10745458  -1.521              0.12833
formatted_work_typePart-time           -5.70188387  2.73655310  -2.084              0.03731 *
formatted_work_typeTemporary           -2.60603664  4.40548656  -0.592              0.55421
clean_salary                           -0.00001796  0.00000550  -3.265              0.00111 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.62 on 2321 degrees of freedom
Multiple R-squared:  0.7594,    Adjusted R-squared:  0.7581
F-statistic: 610.4 on 12 and 2321 DF,  p-value: < 0.00000000000000022
```

As it can be observed, the R-squared value is 0.7581.

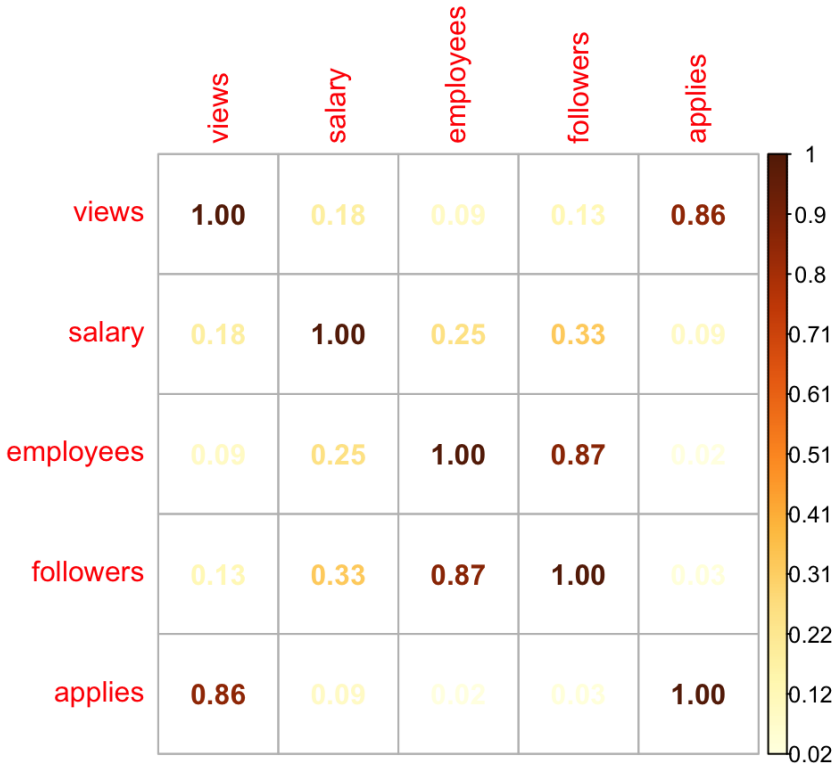
After carrying out Feature Selection, for this model, the performance of the model decreased very slightly. However, it is a good tradeoff, since in the original model, there were around 70 attributes - largely due to different categories in the categorical variables. But now, there are only 12. Feature selection steps reduced the complexity of the model, while the accuracy decreased only very slightly.

2. Filtering, for Neural Networks.

For the Neural Networks, the dependent variable was the number of applicants for a job posting, while the independent variables were the number of views, salary, number of employees working at the company and the number of followers. All were continuous variables.

The data had to be normalized and the best Neural Network model that was plotted consisted of 4 hidden layers, with 3 nodes in the first & second layers, and 2 nodes in the third & fourth layers. The model was then used to predict the number of applicants, based on the independent variables in the testing dataset. The corresponding correlation coefficient value was around 0.912.

For Feature Selection, the Filter method was used. A corrplot was built to visualize the correlation matrix. The training dataset was used.



As it can be observed from the plot above, only the ‘views’ column has a significant correlation value with the dependent variable. All the other variables could be discarded, since their correlation coefficient values are very low. The followers correlation coefficient is 0.87, but it is not relevant to our research question, for the moment, so it was discarded.

The corresponding changes were made in the Neural Networks model. And the correlation coefficient value decreased slightly.

```r
library(neuralnet)
#Only using views column as the IV
applicants_model_main <- neuralnet(formula = applies ~ views, data = applicants_train, hidden = c(3,3,2,2))

model2 = update(applicants_model_main, ~.-salary-employees-followers)
#Plotting the Neural Network Model
plot(applicants_model_main)
model2_main_results <- neuralnet::compute(model2, applicants_test[1])
#applicants_test[1] because the 1st column in the dataframe is 'views' column.
predicted_applicants_model_main <- model2_main_results$net.result
cor(predicted_applicants_model_main,applicants_test$applies)
```

```
        [,1]
[1,] 0.892496
```

Just for reference, A basic Neural Network model was also created, which doesn't contain any hidden layers or nodes. And Feature Selection was applied to it.

```
The code below is for the neural network model, with feature selection, WITHOUT any hidden layers or nodes:
```r
applicants_model_main_2 <- neuralnet(formula = applies ~ views + salary + employees + followers, data =
applicants_train)
model3 = update(applicants_model_main_2, ~.-salary-employees-followers)
#Plotting the Neural Network Model
plot(model3)
model3_main_results <- neuralnet::compute(model3, applicants_test[1])
#applicants_test[1] because the 1st column in the dataframe is 'views' column.
predicted_applicants_model_main_2 <- model3_main_results$net.result
cor(predicted_applicants_model_main_2,applicants_test$applies)
```
```

```
         [,1]
[1,] 0.8946563
```

It was found out that the Correlation Coefficient values are almost similar in both the cases.

Hence, by carrying out the Filter Feature Selection Method, 3 variables could be eliminated. Although the model performance decreased very slightly, the complexity of the model was reduced.

3. Decision Trees:
We employed the Embedded Feature Selection for Decision Trees model, which we used in Milestone 2. It was a Decision Tree classifier model, since the response variable was categorical. And all the predictor variables were categorical too. The data was split into training & testing, the corresponding Decision Tree Model was created and CrossTable generated.

We had also converted all our prominent continuous variables into categorical. They were labelled as class_views, class_applies, class_salary, formatted_work_type, remote_allowed, class_employees, class_followers.

After creating the Confusion Matrix for the testing dataset and the predicted model, from the overall statistics, we get an accuracy value of around 0.72 or 0.71.

```
Confusion Matrix and Statistics

          Reference
Prediction High  Low Medium
    High    731   57    319
    Low       9 1050    148
    Medium   56  416    829

Overall Statistics

               Accuracy : 0.722
                 95% CI : (0.7071, 0.7366)
    No Information Rate : 0.4213
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5806

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: High Class: Low Class: Medium
Sensitivity               0.9183     0.6894        0.6397
Specificity               0.8666     0.9250        0.7965
Pos Pred Value            0.6603     0.8699        0.6372
Neg Pred Value            0.9741     0.8036        0.7982
Precision                 0.6603     0.8699        0.6372
Recall                    0.9183     0.6894        0.6397
F1                        0.7683     0.7692        0.6384
Prevalence                0.2202     0.4213        0.3585
Detection Rate            0.2022     0.2905        0.2293
Detection Prevalence      0.3062     0.3339        0.3599
Balanced Accuracy         0.8925     0.8072        0.7181
```

For Embedded FS, we made use of Random Forests. A randomForest model was created in R, by using the training dataset. The response variable was the class of the number of applicants & all the remaining class variables were predictor variables.

The importance() function was then used to find out the MeanDecreaseGini values for each of the predictor variables.

```
#Creating the Random Forest Model:
random_forest_model <- randomForest(class_applies ~ ., data = train)
importance(random_forest_model)

```

```
                     MeanDecreaseGini
class_views                1848.04209
class_salary                 99.35081
formatted_work_type         125.10113
remote_allowed              165.84459
class_employees             106.16312
class_followers              41.74513
```

As we can observe, the MeanDecreaseGini value is much significantly higher than all the other variables. Hence, it is the most prominent one and other variables can be discarded.

We then created a new Decision Tree model, with only the class_views column as the predictor, after splitting into training and testing datasets

```
```{r}
#Creating new decision tree model, with only the class_views as the predictor

      applies_model_2 <- C5.0(train[1], train$class_applies)
      summary(applies_model)

      #Predicting the values in the testing dataset
      applies_predictions_2 <- predict(applies_model_2, test)

      #Creating the CrossTable
              CrossTable(test$class_applies, applies_predictions_2,
                 prop.chisq = FALSE, prop.c = FALSE,
                 prop.r = FALSE,
                 dnn = c('actual applications', 'predicted applications'))

#Getting the Overall Statistics value
confusionMatrix(test$class_applies, applies_predictions_2, mode="everything")
```
```

```
Confusion Matrix and Statistics

          Reference
Prediction High Low Medium
    High    707 110    290
    Low       0 863    344
    Medium   35 271    995

Overall Statistics

               Accuracy : 0.7095
                 95% CI : (0.6944, 0.7243)
    No Information Rate : 0.4506
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.56

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: High Class: Low Class: Medium
Sensitivity               0.9528     0.6937        0.6108
Specificity               0.8608     0.8549        0.8459
Pos Pred Value            0.6387     0.7150        0.7648
Neg Pred Value            0.9860     0.8418        0.7260
Precision                 0.6387     0.7150        0.7648
Recall                    0.9528     0.6937        0.6108
F1                        0.7647     0.7042        0.6792
Prevalence                0.2053     0.3441        0.4506
Detection Rate            0.1956     0.2387        0.2752
Detection Prevalence      0.3062     0.3339        0.3599
Balanced Accuracy         0.9068     0.7743        0.7284
```

It can be observed that the value of Accuracy came out around 0.7095. It slightly decreased from the previous Model. However, even though the performance decreased slightly, the model has become less complex now. In the first model, it had 6 variables, but now, using only 1 variable after using feature selection provides an almost similar value for the accuracy.

Question 6. Extra Credit

Discuss potential ethical issues in the research questions you have proposed for your project. As covered in Class 10 (Ethical Data Science part), delve into concerns around data collection, variable manipulation and model evaluation:

There are a few potential Ethical Issues in the research questions we have proposed for our project:

1. Data Collection:

   The dataset covers job postings from LinkedIn over a short period of 2 days. This might lead to biases in the data due to its limited timeframe, potentially not representing the entire spectrum of job postings throughout the year. This could result in skewed predictions, as certain job sectors or regions might be over- or underrepresented. Certain industries only recruit during specific times of the year. In addition, certain industries may not use LinkedIn as a method of applicant collection.

   Also, most importantly, the dataset was acquired from the website Kaggle, where the original creator of this dataset used a web scraping tool - developed in Python - to obtain the data. It is essential to consider if the data collection had explicit consent or agreement from LinkedIn or the job posting companies whose data was scraped.

   There is also bias in that some of the jobs do not show the number of applicants at all, since the posting sends applicants directly to their corporate website.

   In addition, since these applicants were all online, there is a bias against those who do not have access to the internet at home or on a mobile device.

2. Privacy:

   These job postings may contain sensitive information, such as the contact details of the recruiters. It is important to ensure that the dataset is anonymized and does not reveal personally identifiable information to prevent potential privacy breaches. While the dataset might not contain explicitly identifiable personal information, it still includes job-related details that might indirectly identify individuals or specific companies.

3. Variable Manipulation and Implications:

   Manipulating variables or using certain attributes (e.g., salary, location) could lead to reinforcing biases or perpetuating stereotypes.

For instance, if certain job types or locations are implicitly associated with particular demographics, it could lead to biased predictions or reinforce discriminatory practices. One common example is that most locations for jobs in the Information Technology field will be in San Francisco. Or jobs that required either a particular citizenship status would be found near Washington D.C.

Variables like "number of employees" or "company description" may inadvertently include biases against certain company sizes or types.

Since one of our research questions is related to the number of applicants for a particular job posting, companies could inflate that value, to keep up appearances and show that the company is in demand. Alternatively, for some companies the number of applicants is not available at all.

Another example would be related to the number of employees and the salary of that company. It could represent a parabolic relationship. The more the number of employees working at the company, the lower could be the mean salary of the employees. In addition, certain types of roles may be listed in LinkedIn, but not all types.

4. Unintended Bias in Location:

   Variables related to the location might inadvertently discriminate against applicants from certain regions or ethnic groups. For instance, if the dataset predominantly features job postings from affluent areas, such as Los Angeles or San Francisco, the model might inadvertently favor applicants from those regions, leading to biased predictions against applicants from less privileged areas. Applicants would have second thoughts about working in a company where the cost of living would be much higher, if remote work is not allowed.

5. Implications of Model Use:

   We would also have to consider how the predictive models we developed for all the Milestones might be used in practice. If it influences hiring decisions or resource allocation, biases in the model could perpetuate inequalities or reinforce existing systemic issues in employment practices.

Contributions

Sandy Staub wrote the code for and answered question 1 in the report, created the relevant slides and recorded those sections. Erik Rye wrote the code for and answered questions 3 and 4 in the report, as well as created the slides covering their content and recording those sections. Sharvil Shastri wrote the code for and answered questions 2 & 5 in the report, as well as created the corresponding slides & recorded the video for those sections.
Sharvil and Sandra also worked on Q6 - Extra Credit.

Sharvil compiled all the video recordings and uploaded the final video presentation on Youtube