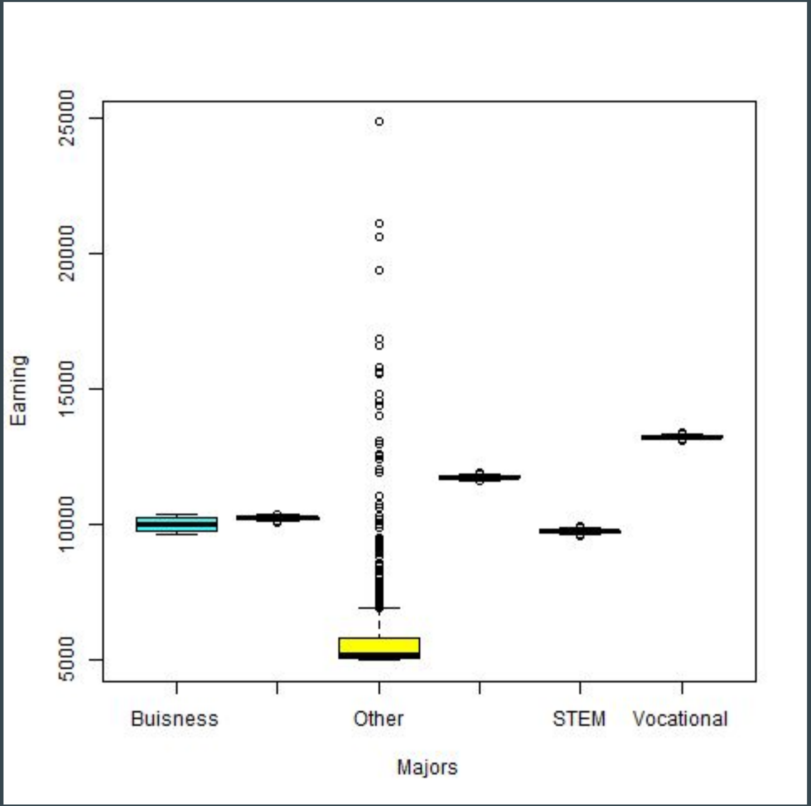
Earning Predicting Model

...

By: Rohit Manjunath

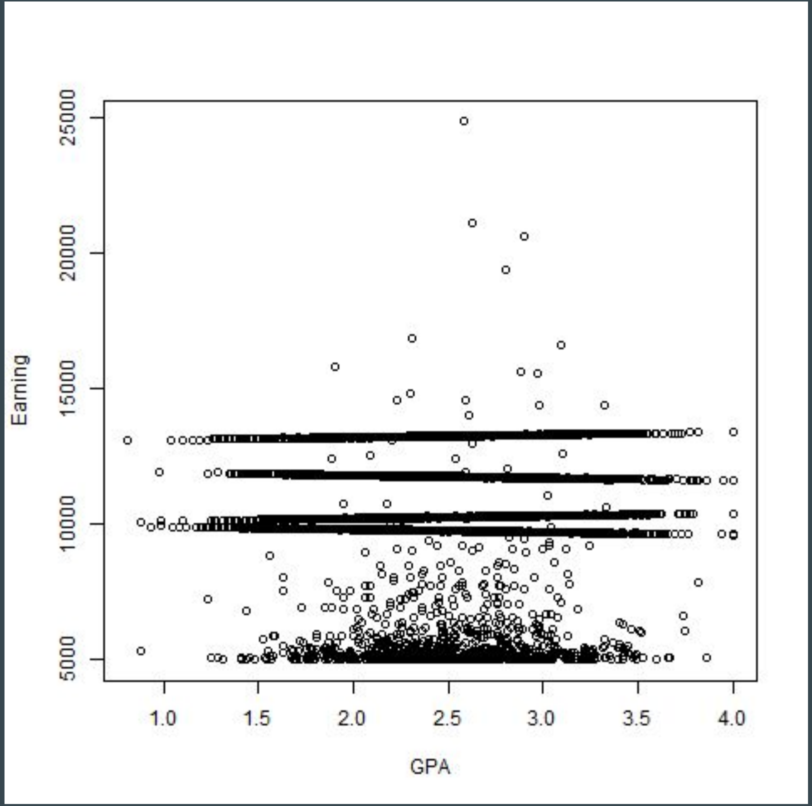
Major and Earnings

- As you can see in the box graph on the right, all majors have a very small range except ‘other’ and do not tend to vary much.
- So major is a pretty easy way to narrow someone’s income into a range. Unless its ‘other’.



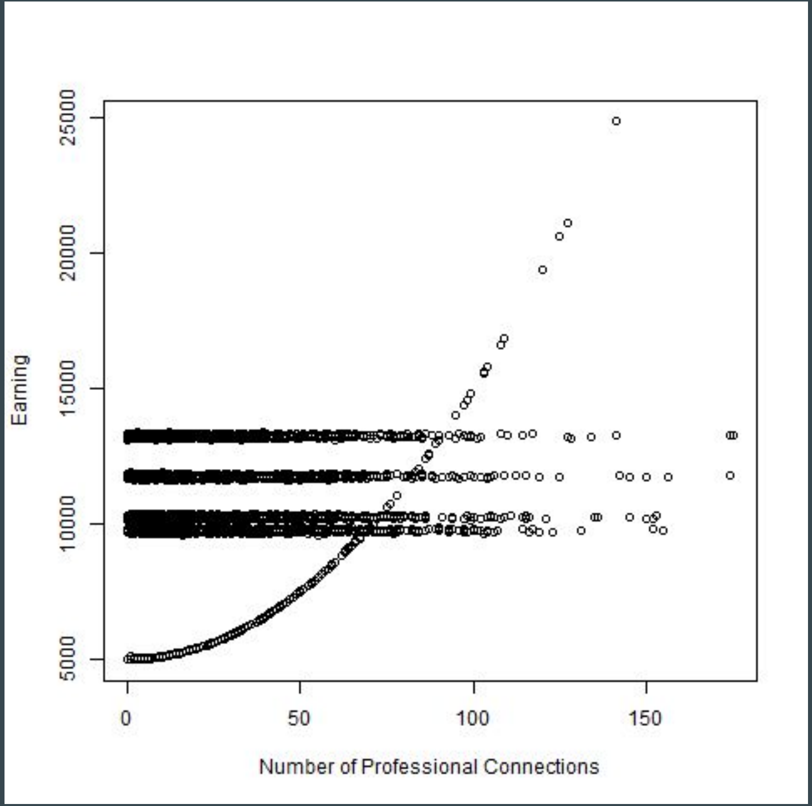
GPA and Earnings

- As you can see in the scatter plot on the right, there are weird ‘trend lines’ formed with gpa.
- This shows that GPA too affects earnings in a weird way.



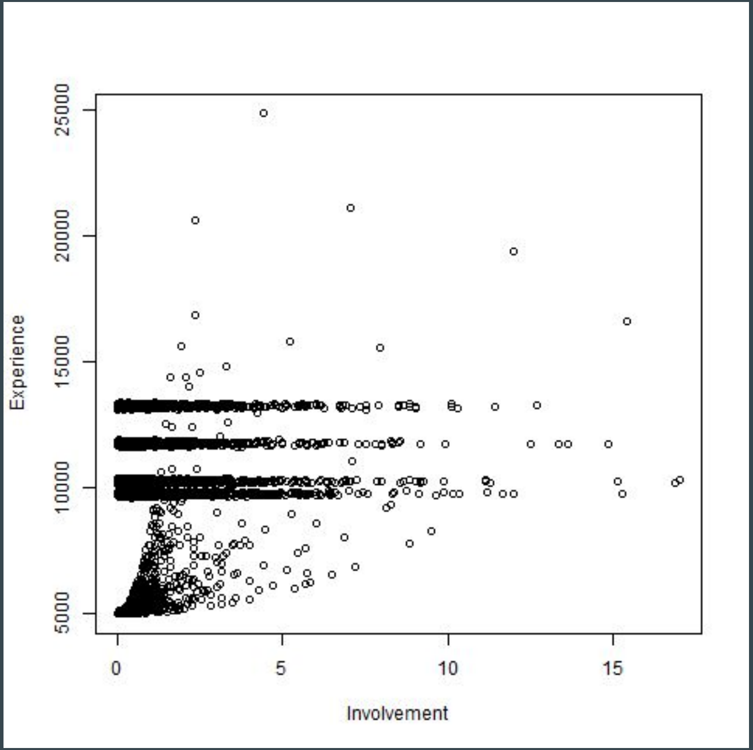
Number of Professional Connections and Earnings

- As you can see in the scatter plot on the right, there are 4 different trends formed with professional connection.
- One upward sloping curve.
- Four horizontal lines.
- Breaking this further down into certain majors(subsetting by major and plotting the same graph on the right) show that each trend line is associated with a different major.



New Columns | Involvement

- Year of graduation alone is useless since it doesn't tell us much.
- However, we can use that data to see how active the person is amongst the 'working community'. To see how engaged the person is in the working community we can divide number of connections by number of years of experience.
- Number of years of experience is 2021 - graduation year.
- The graph on the right is similar to the previous graph.



New Columns | Major Into Number

- Major is a categorical column. Machine learning models such as linear regression require numbers to find the best fit line. Hence, to convert it into numerical values we can manually encode it.

  Code:

  ```
  decision <- rep(0,nrow(earnings))

  decision[earnings$Major == "Humanities"] <- 1

  decision[earnings$Major == "Vocational"] <- 2

  decision[earnings$Major == "Professional"] <- 3

  decision[earnings$Major == "Buisness"] <- 4

  decision[earnings$Major == "Other"] <- 5

  earnings$MajorNum <- decision
  ```

- As you can see we create a new column based on the Major by assigning numbers from 0 to 5.

Linear Regression

- Using the linear regression model let us pass in the parameters and the predicting column and see how it fares by calculating it's error(mean squared error).
- Code:

```
trainLm <- lm(Earnings ~ GPA + MajorNum + Height + Number_Of_Parking_Tickets +
Number_Of_Credits + Involvement,  data = train)

predLm <-  predict(trainLm, newdata = test)

error <- mean((predLm - test$Earnings)^2)
```

- The mean square error or 'error' averages around 200,000. Taking the square root of that we get 447. Hence, our prediction's mean is off by 447 dollars(Earnings). We can get a better model. Let's keep this as the base.

Decision Tree

- Using the decision tree model let us pass in the parameters and the predicting column and see how it fares by calculating it's error(mean squared error).
- Code:

```
tree <- rpart(Earnings ~ GPA + Number_Of_Professional_Connections + Major + Graduation_Year + Height +
Number_Of_Credits + Number_Of_Parking_Tickets, data = train, method = "anova")

rpart.plot(tree)

test$NewEarnings <- predict(tree, newdata = test)

error <- mean((test$NewEarnings - test$Earnings)^2)
```

- The mean square error or 'error' averages around 90,000. Taking the square root of that we get 300. Hence, our prediction's mean is off by 300 dollars(Earnings). This looks like its a better model than our linear regression model. Can we get it lower?

Random Forest

- Using the random forest model let us pass in the parameters and the predicting column and see how it fares by calculating it’s error(mean squared error).
- Code:

```
trainRf <- randomForest(Earnings ~ GPA + Number_Of_Professional_Connections + Major +
Graduation_Year + Height + Number_Of_Credits + Number_Of_Parking_Tickets,  data = train)

predRf <-  predict(trainRf, newdata = test)

error <- mean((predRf - test$Earnings)^2)
```

- The mean square error or ‘error’ averages around 95,000. It was always a bit higher than our decision tree model. Taking the square root of MSE we get 308. Hence, our prediction’s mean is off by 308 dollars(Earnings). This could not beat our decision tree model.

SVM

- Using the svm model let us pass in the parameters and the predicting column and see how it fares by calculating it's error(mean squared error).
- Code:

```
trainSvm = svm(Earnings ~ GPA + Number_Of_Professional_Connections + Major + Height +
Number_Of_Parking_Tickets + Number_Of_Credits + Graduation_Year, data = train)

predSvm <-  predict(trainSvm, newdata = test)

error <- mean((predSvm - test$Earnings)^2)
```

- The mean square error or 'error' averages around 50,000. It was always a bit higher than our decision tree model. Taking the square root of MSE we get 224. Hence, our prediction's mean is off by 224 dollars(Earnings). This does beat your decision tree model.

Conclusion

- We can conclude that out of the all the models tried, SVM gives the best result as MSE is the closest to 0.

Thank you!