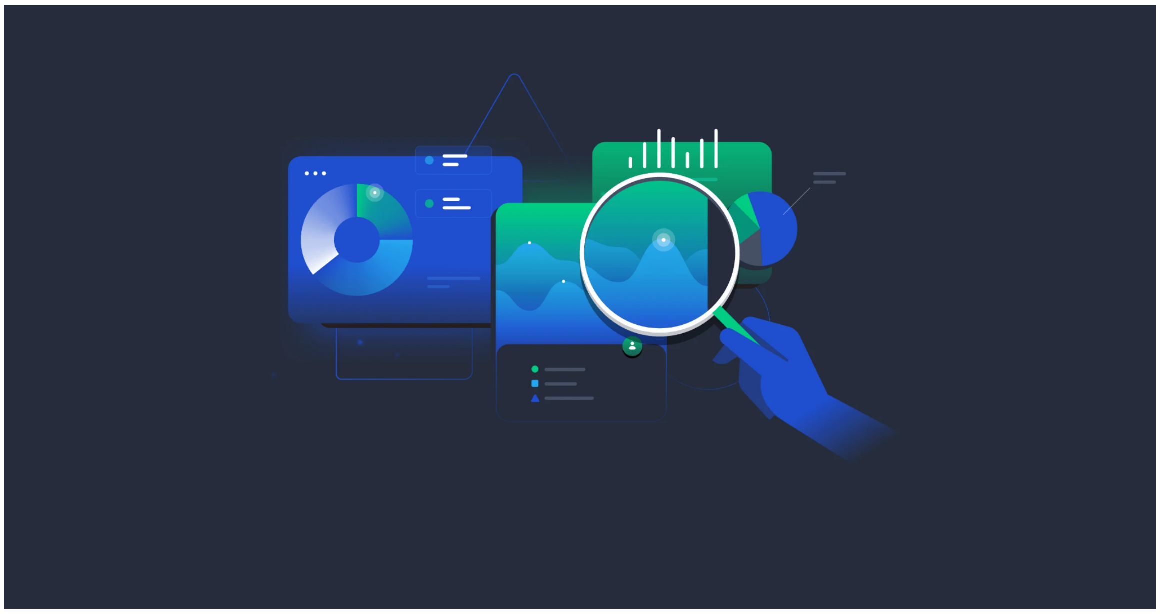


Modeling Data Science Salaries

STAT 574 FINAL PROJECT

Submitted to
Dr. Olga Korosteleva

Report Prepared By:
Dana Villegas

May 3, 2023

Table of Contents

Introduction..2

Background..2

Data Description..3

Results..3

Conclusion..4

Appendix..5

A. Random Forest..5

i. R Code..5

ii. R Output..6

iii. Python Code..7

iv. Python Output..7

B. XGBoost..8

i. R Code..8

ii. R Output..9

iii. Python Code..9

iv. Python Output..10

C. Text Mining..10

i. R Code..10

ii. R Output..12

References..15

Ⅰ. Introduction

Data Science is a captivating and fast-growing career field. Extracting and analyzing data to produce meaningful conclusions is an important part of any business. As an aspiring data analyst graduating in less than a month, I thought it would be very insightful to base my final project for data mining on job postings specifically for Data Science within the US. This way, I can examine what companies believe are the most important qualities in candidates looking to break into Data Science and it could help me on my job hunt when I graduate. By training models using Random Forest and XGBoost on data linking data scientists' salaries, seniority, skill-set, and more I hope to uncover the most important determinants of a data scientist's salary. Moreover, I use Text Mining to provide job-seekers in the field with more information on which companies and locations they should look into.

ⅠⅠ. Background

Data science has evolved rapidly, advancing from a niche field combining machine learning and statistics, to a broad, cross-disciplinary area with applications across every industry. The ability to collect and analyze massive amounts of data has enabled new discoveries and insights that were not possible before. Using data science techniques, businesses gain a huge competitive advantage because these techniques allow them to optimize key metrics and improve critical decision making. Because the field is evolving rapidly, the demand for data science jobs has grown tremendously in recent years, and is expected to continue increasing for the foreseeable future.

As the volume of data in the world continues to grow, so does the need for people who can analyze it. The strong demand and the inability to meet demand has led to rising salaries within the data science field. With fierce competition for these coveted and well-paying jobs, it is crucial for candidates to go above and beyond in their skillset. This skill set includes, but is not limited to: fluency in one or more programming languages, Machine Learning, statistics, problem-solving, data visualization and data mining.

ⅠⅠⅠ. Data Description

My dataset was pulled from Kaggle, containing 2084 rows and 23 columns. Some of the variables included were company rating, job type, and seniority level with the response variable being salary estimate. Since there were many columns of the string type, I decided to split my dataset in two; one with the numeric variables to implement my machine learning techniques and the other with the strings to utilize text mining on. After importing the data I omitted a few columns such as company sector and company size just because I felt the other predictors might've held more importance. I saw there were a few NA values, so I omitted them, leaving me with about 1300 observations to work with. I then built my random forest and XGBoost models with the following as predictors: hourly, seniority, company_age, python_yn, spark_yn, azure_yn, aws_yn, excel_yn, machine_learning_yn, description length, job_simpl and rating.

ⅠV. Results

After building my random forest model in R and adjusting the parameters accordingly, I achieved prediction accuracies of 46%, 65%, and 78% within 10, 15 and 20 percent of the testing data respectively. With Python, I reached accuracies of 82%, 87% and 89% respectively, which was exponentially better than in R. With XGBoost, I was able to increase the prediction accuracies in R to 81%, 85% and 89% respectively with 10, 15, and 20 percent of the testing set. This was a huge, yet expected improvement since XGBoost is known to boost a model's accuracy and enhance its performance. In Python, the accuracies came out to 84%, 88% and 91% after implementing XGBoost, improving the model even more than before.

Once the models were built and the accuracies were calculated, I explored the variable importance within each model. For my Random Forest model in R, the top 5 variables having the most significant impact were company age, job_simpl which refers to job type, job description length, company rating, and seniority. The only predictor I was a bit stunned to see in the top 5 was job description, however after taking some time to rationalize it, I discovered that it makes sense. If a company is willing to pay a lot for a data scientist, they will be specific in what they want and therefore, they will write a lengthier, detailed job description for the ideal candidate. In Python, the variable importance was the same for the most part, with just a slight difference in order. For XGBoost in R and Python, the variables in the top 5 stayed the same as well, with a slight variation in order.

After my models were built, I created another dataset containing all the columns with strings from the original dataset. I pulled the following variables: Company, Location, Job Description, Job Type, and Salary Estimate. Using this new dataset, I filtered the data to only include a high salary estimate, which I defined as $90,000 and above. My goal here was to discover what companies and locations had the most data science jobs paying a high salary so aspiring students in the field could have leads on which companies to look into if they want a high paying job in data science. I also analyzed the job descriptions to see if any words stood out in terms of specific skills and/or qualities companies are looking for, especially for a high paying job in data science. Below are the visuals I created after implementing text mining.

V. Conclusion

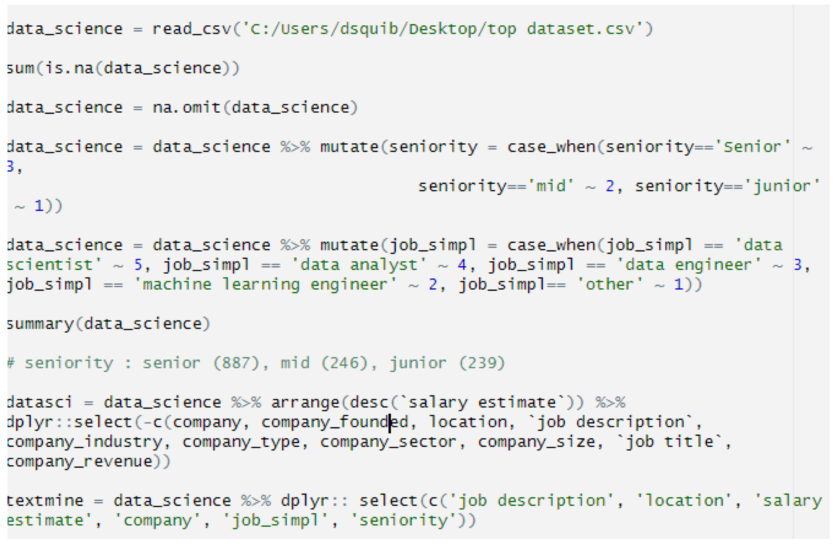
Overall, when building machine learning models to predict data science salaries, it is highly recommended to use Python over R, since Python produced the highest prediction accuracies for both the Random Forest and XGBoost models. I also discovered company age is the most important feature in both languages and models. This makes sense since a company that is older will have had more time to build a reputation for itself, gain more experience in the industry, and generate a revenue more so than a startup. Therefore it is highly likely an older company will pay their data scientists more generously as well. The type of job, description length, company rating and seniority were also some of the most important features across both models and languages, so it is important to consider these aspects when looking for a high paying job in this field.

I learned through the implementation of text mining that the company with the most data science related jobs paying a high salary is Tiktok, which does not come as a surprise given they are a well-known social media platform always in need of data engineers and scientists. The location with the most jobs paying a high salary for data science is Remote, meaning you don’t actually have to leave your home in order to find a well paying job in the field. Given the events over the past few years, this makes sense as more and more companies are letting people work from the comfort of their home rather than come into the office everyday. With all the discoveries I’ve made through this project, I have unlocked valuable information that I will be able to keep in mind when job hunting in the future, and I hope my findings prove useful to other aspiring data scientists in the future.

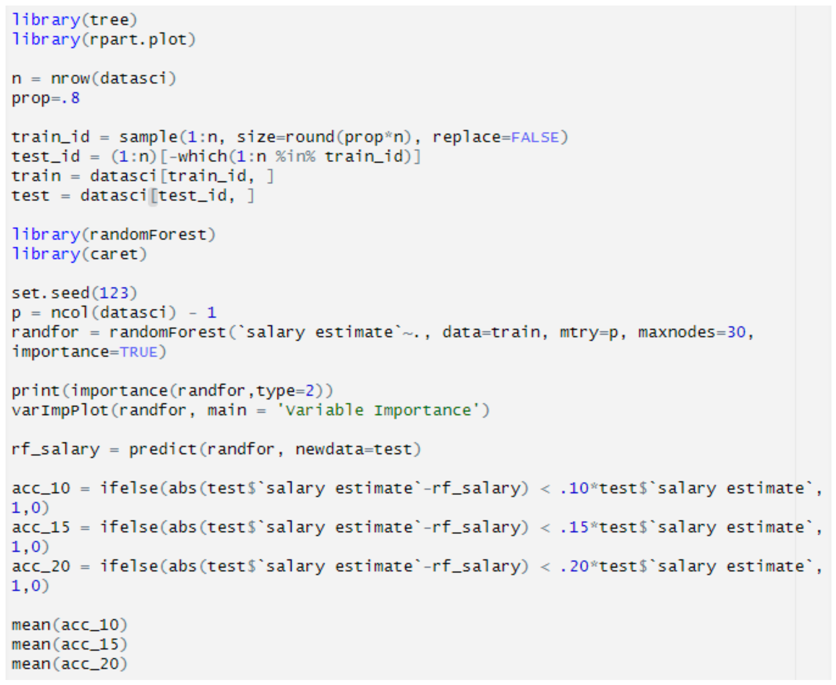
VⅠ. Appendix

A. Random Forest

(a) R Code



```r
data_science = read_csv('C:/Users/dsquib/Desktop/top dataset.csv')

sum(is.na(data_science))

data_science = na.omit(data_science)

data_science = data_science %>% mutate(seniority = case_when(seniority=='Senior' ~
3,
                                             seniority=='mid' ~ 2, seniority=='junior'
 ~ 1))

data_science = data_science %>% mutate(job_simpl = case_when(job_simpl == 'data
scientist' ~ 5, job_simpl == 'data analyst' ~ 4, job_simpl == 'data engineer' ~ 3,
job_simpl == 'machine learning engineer' ~ 2, job_simpl== 'other' ~ 1))

summary(data_science)

# seniority : senior (887), mid (246), junior (239)

datasci = data_science %>% arrange(desc(`salary estimate`)) %>%
dplyr::select(-c(company, company_founded, location, `job description`,
company_industry, company_type, company_sector, company_size, `job title`,
company_revenue))

textmine = data_science %>% dplyr:: select(c('job description', 'location', 'salary
estimate', 'company', 'job_simpl', 'seniority'))
```



```r
library(tree)
library(rpart.plot)

n = nrow(datasci)
prop=.8

train_id = sample(1:n, size=round(prop*n), replace=FALSE)
test_id = (1:n)[-which(1:n %in% train_id)]
train = datasci[train_id, ]
test = datasci[test_id, ]

library(randomForest)
library(caret)

set.seed(123)
p = ncol(datasci) - 1
randfor = randomForest(`salary estimate`~., data=train, mtry=p, maxnodes=30,
importance=TRUE)

print(importance(randfor,type=2))
varImpPlot(randfor, main = 'Variable Importance')

rf_salary = predict(randfor, newdata=test)

acc_10 = ifelse(abs(test$`salary estimate`-rf_salary) < .10*test$`salary estimate`,
1,0)
acc_15 = ifelse(abs(test$`salary estimate`-rf_salary) < .15*test$`salary estimate`,
1,0)
acc_20 = ifelse(abs(test$`salary estimate`-rf_salary) < .20*test$`salary estimate`,
1,0)

mean(acc_10)
mean(acc_15)
mean(acc_20)
```

Figures 1 & 2: R Code for Random Forest Model

(b) R Output

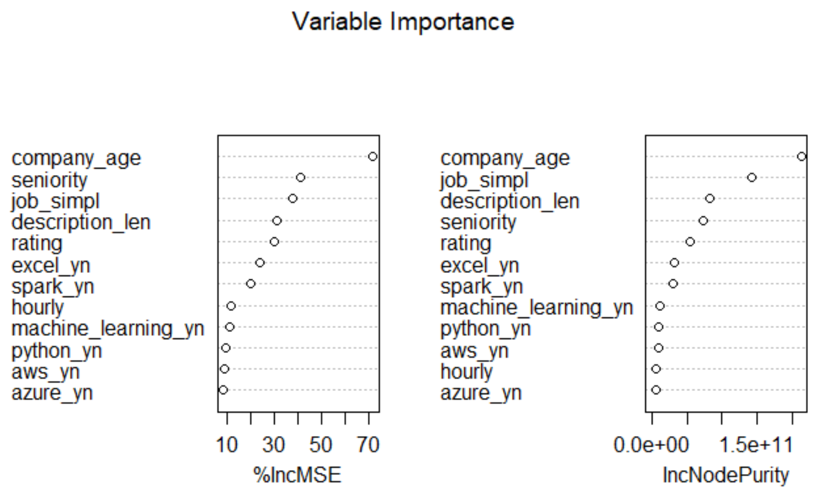
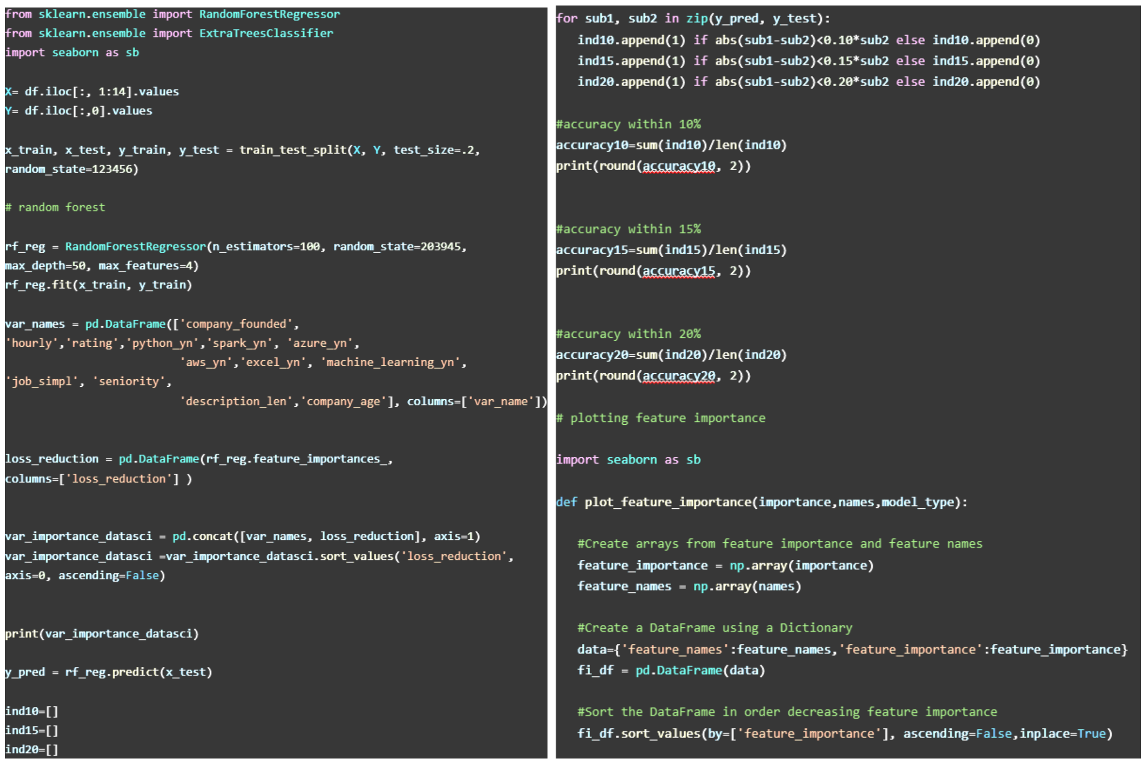


Figure 3: R Output for Random Forest Variable Importance

```
                    IncNodePurity
hourly                 5950947059
rating                53960229168
python_yn             10319219055
spark_yn              30718331844
azure_yn               5368447210
aws_yn                 9874942439
excel_yn              31497903262
machine_learning_yn   11932877406
job_simpl            141339687303
seniority             72954882915
description_len       81847957159
company_age          212625055352
[1] 0.5291971
[1] 0.649635
[1] 0.7335766
```

Figure 4: R Output for Random Forest Variable Importance and prediction accuracies within 10, 15 and 20 percent of the testing data.

(c) Python Code

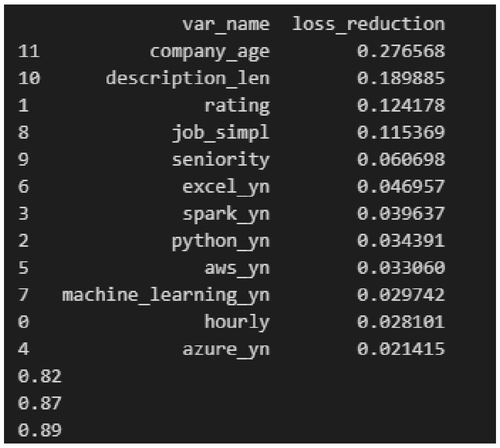


```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier
import seaborn as sb

X= df.iloc[:, 1:14].values
Y= df.iloc[:,0].values

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=.2,
random_state=123456)

# random forest

rf_reg = RandomForestRegressor(n_estimators=100, random_state=203945,
max_depth=50, max_features=4)
rf_reg.fit(x_train, y_train)

var_names = pd.DataFrame(['company_founded',
'hourly','rating','python_yn','spark_yn', 'azure_yn',
                          'aws_yn','excel_yn', 'machine_learning_yn',
'job_simpl', 'seniority',
                          'description_len','company_age'], columns=['var_name'])


loss_reduction = pd.DataFrame(rf_reg.feature_importances_,
columns=['loss_reduction'] )


var_importance_datasci = pd.concat([var_names, loss_reduction], axis=1)
var_importance_datasci =var_importance_datasci.sort_values('loss_reduction',
axis=0, ascending=False)


print(var_importance_datasci)

y_pred = rf_reg.predict(x_test)

ind10=[]
ind15=[]
ind20=[]
```

```
for sub1, sub2 in zip(y_pred, y_test):
   ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
   ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
   ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(round(accuracy10, 2))


#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(round(accuracy15, 2))


#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print(round(accuracy20, 2))

# plotting feature importance

import seaborn as sb

def plot_feature_importance(importance,names,model_type):

   #Create arrays from feature importance and feature names
   feature_importance = np.array(importance)
   feature_names = np.array(names)

   #Create a DataFrame using a Dictionary
   data={'feature_names':feature_names,'feature_importance':feature_importance}
   fi_df = pd.DataFrame(data)

   #Sort the DataFrame in order decreasing feature importance
   fi_df.sort_values(by=['feature_importance'], ascending=False,inplace=True)
```

Figure 5: Python Code for Random Forest Model

(d) Python Output



```
               var_name  loss_reduction
11          company_age        0.276568
10      description_len        0.189885
1                rating        0.124178
8             job_simpl        0.115369
9             seniority        0.060698
6              excel_yn        0.046957
3              spark_yn        0.039637
2             python_yn        0.034391
5                aws_yn        0.033060
7   machine_learning_yn        0.029742
0                hourly        0.028101
4              azure_yn        0.021415
0.82
0.87
0.89
```

Figure 6: Python Output of Loss Reduction and prediction accuracies within 10, 15, and 20 percent of the testing data.

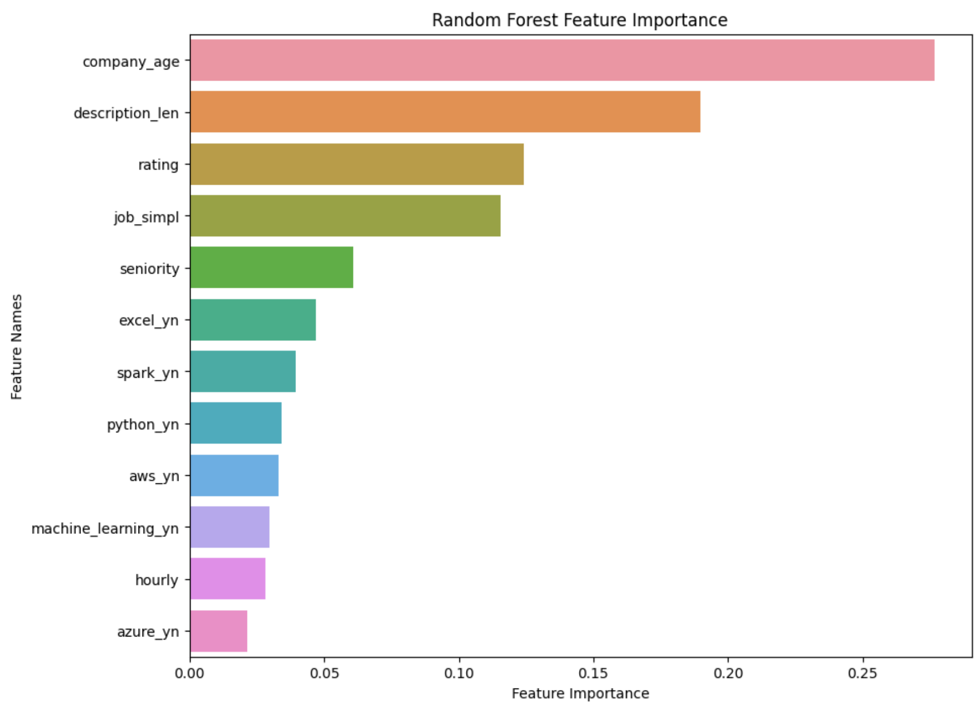
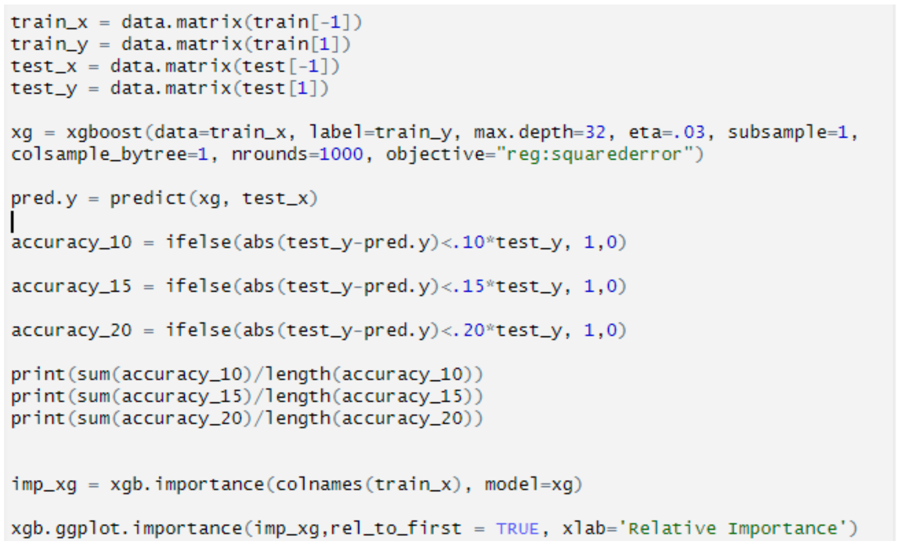


Figure 7: Python graph of the variable importance for Random Forest model

B. XGBoost Model

(a) R Code



```
train_x = data.matrix(train[-1])
train_y = data.matrix(train[1])
test_x = data.matrix(test[-1])
test_y = data.matrix(test[1])

xg = xgboost(data=train_x, label=train_y, max.depth=32, eta=.03, subsample=1,
colsample_bytree=1, nrounds=1000, objective="reg:squarederror")

pred.y = predict(xg, test_x)

accuracy_10 = ifelse(abs(test_y-pred.y)<.10*test_y, 1,0)

accuracy_15 = ifelse(abs(test_y-pred.y)<.15*test_y, 1,0)

accuracy_20 = ifelse(abs(test_y-pred.y)<.20*test_y, 1,0)

print(sum(accuracy_10)/length(accuracy_10))
print(sum(accuracy_15)/length(accuracy_15))
print(sum(accuracy_20)/length(accuracy_20))


imp_xg = xgb.importance(colnames(train_x), model=xg)

xgb.ggplot.importance(imp_xg,rel_to_first = TRUE, xlab='Relative Importance')
```

Figure 8: R Code for XGBoost model

(b) R Output

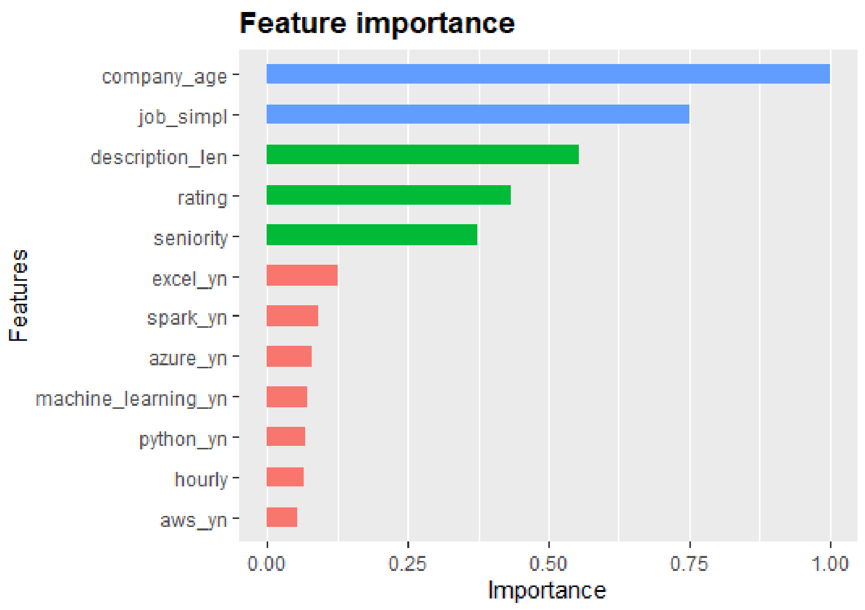
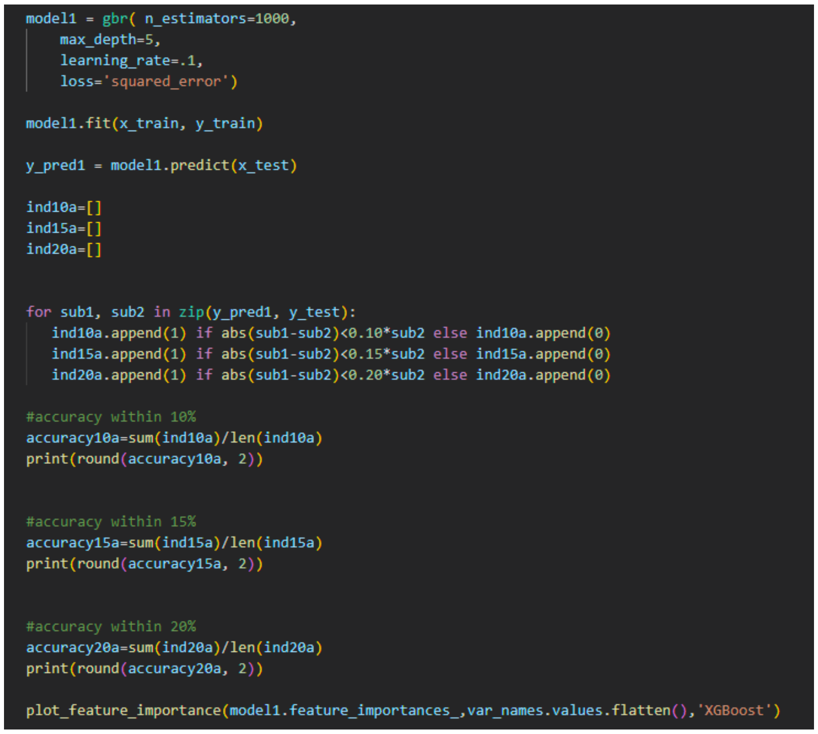


Figure 9: R graph of the variable importance for XGBoost model

(c) Python Code



```
model1 = gbr( n_estimators=1000,
    max_depth=5,
    learning_rate=.1,
    loss='squared_error')

model1.fit(x_train, y_train)

y_pred1 = model1.predict(x_test)

ind10a=[]
ind15a=[]
ind20a=[]


for sub1, sub2 in zip(y_pred1, y_test):
   ind10a.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10a.append(0)
   ind15a.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15a.append(0)
   ind20a.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20a.append(0)

#accuracy within 10%
accuracy10a=sum(ind10a)/len(ind10a)
print(round(accuracy10a, 2))


#accuracy within 15%
accuracy15a=sum(ind15a)/len(ind15a)
print(round(accuracy15a, 2))


#accuracy within 20%
accuracy20a=sum(ind20a)/len(ind20a)
print(round(accuracy20a, 2))

plot_feature_importance(model1.feature_importances_,var_names.values.flatten(),'XGBoost')
```

Figure 10: Python code for XGBoost model

(d) Python Output

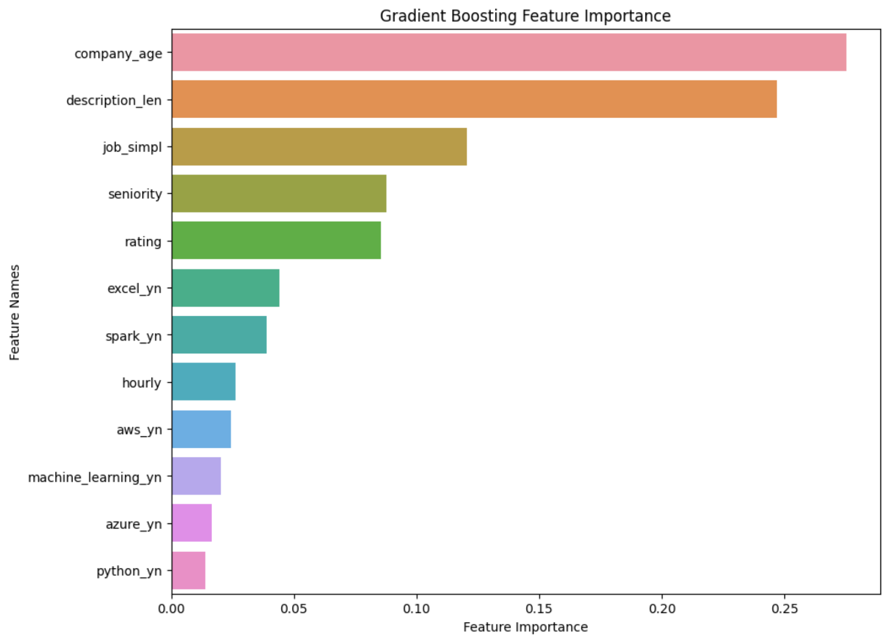
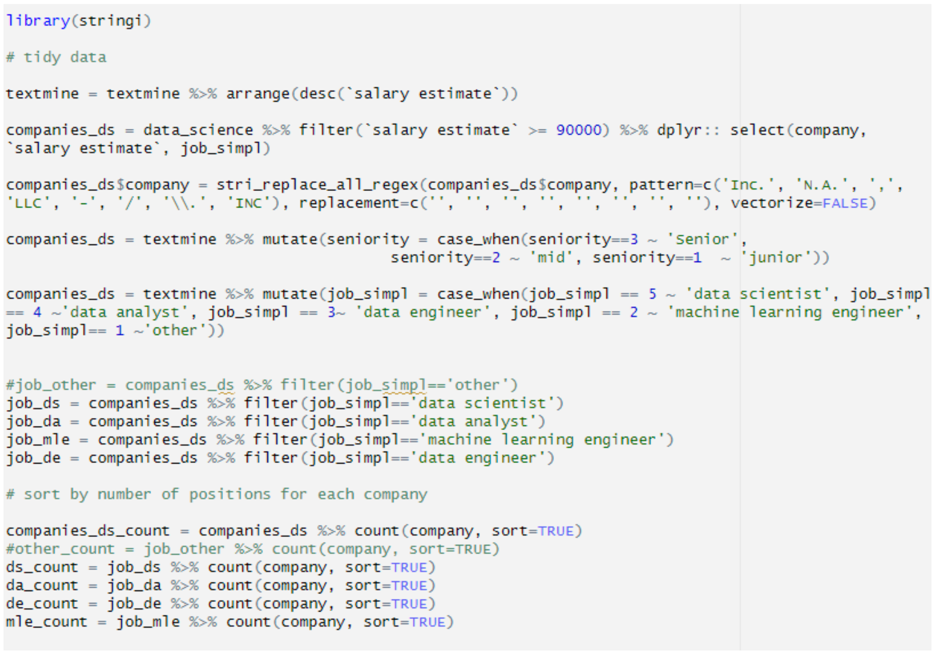


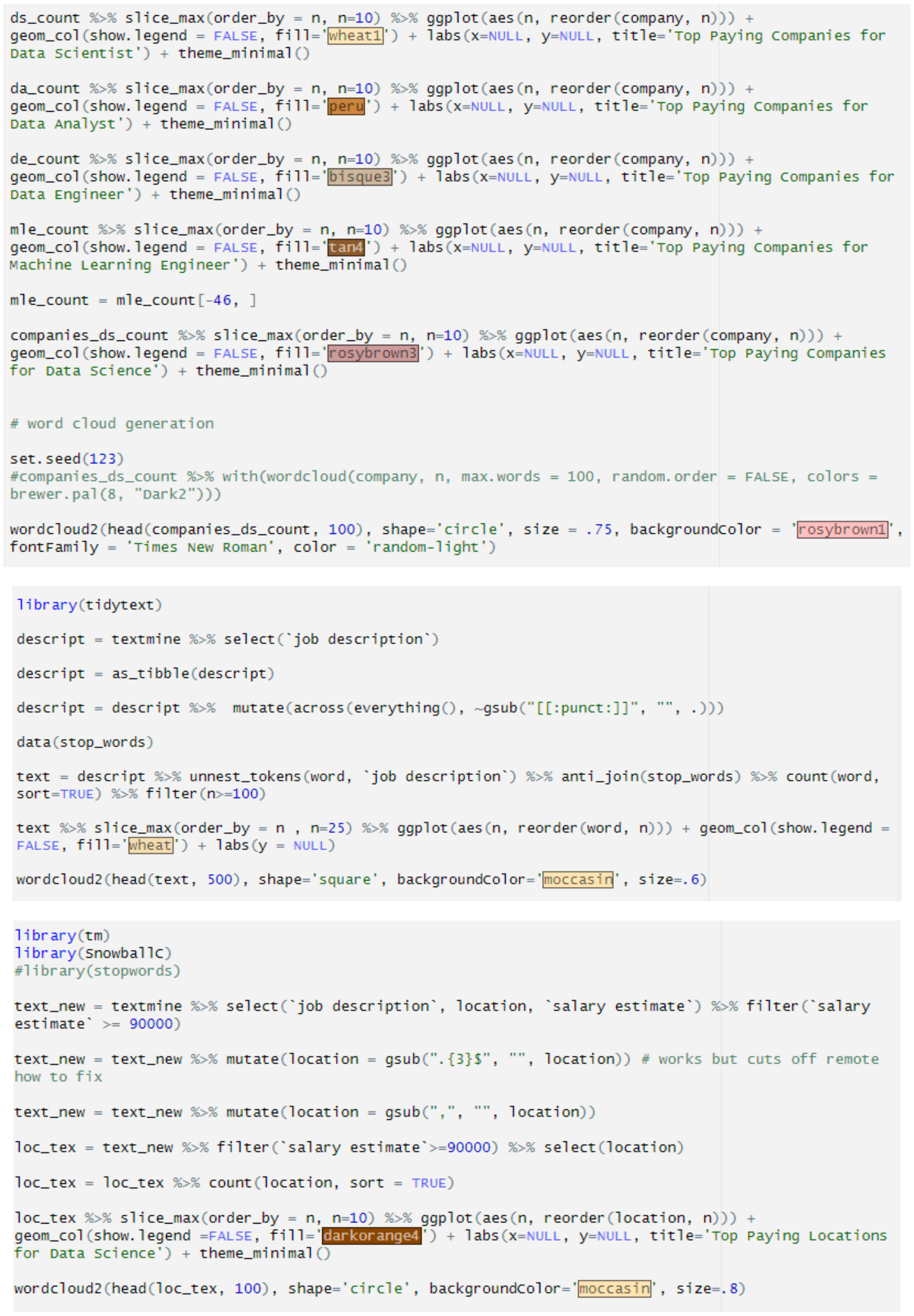
Figure 11: Python graph of the variable importance for Gradient Boosting model

C. Text Mining

(a) R Code



```
library(stringi)

# tidy data

textmine = textmine %>% arrange(desc(`salary estimate`))

companies_ds = data_science %>% filter(`salary estimate` >= 90000) %>% dplyr:: select(company,
`salary estimate`, job_simpl)

companies_ds$company = stri_replace_all_regex(companies_ds$company, pattern=c('Inc.', 'N.A.', ',',
'LLC', '-', '/', '\\.', 'INC'), replacement=c('', '', '', '', '', '', '', ''), vectorize=FALSE)

companies_ds = textmine %>% mutate(seniority = case_when(seniority==3 ~ 'Senior',
                                       seniority==2 ~ 'mid', seniority==1  ~ 'junior'))

companies_ds = textmine %>% mutate(job_simpl = case_when(job_simpl == 5 ~ 'data scientist', job_simpl
== 4 ~'data analyst', job_simpl == 3~ 'data engineer', job_simpl == 2 ~ 'machine learning engineer',
job_simpl== 1 ~'other'))


#job_other = companies_ds %>% filter(job_simpl=='other')
job_ds = companies_ds %>% filter(job_simpl=='data scientist')
job_da = companies_ds %>% filter(job_simpl=='data analyst')
job_mle = companies_ds %>% filter(job_simpl=='machine learning engineer')
job_de = companies_ds %>% filter(job_simpl=='data engineer')

# sort by number of positions for each company

companies_ds_count = companies_ds %>% count(company, sort=TRUE)
#other_count = job_other %>% count(company, sort=TRUE)
ds_count = job_ds %>% count(company, sort=TRUE)
da_count = job_da %>% count(company, sort=TRUE)
de_count = job_de %>% count(company, sort=TRUE)
mle_count = job_mle %>% count(company, sort=TRUE)
```



```r
ds_count %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(company, n))) +
geom_col(show.legend = FALSE, fill='wheat1') + labs(x=NULL, y=NULL, title='Top Paying Companies for
Data Scientist') + theme_minimal()

da_count %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(company, n))) +
geom_col(show.legend = FALSE, fill='peru') + labs(x=NULL, y=NULL, title='Top Paying Companies for
Data Analyst') + theme_minimal()

de_count %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(company, n))) +
geom_col(show.legend = FALSE, fill='bisque3') + labs(x=NULL, y=NULL, title='Top Paying Companies for
Data Engineer') + theme_minimal()

mle_count %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(company, n))) +
geom_col(show.legend = FALSE, fill='tan4') + labs(x=NULL, y=NULL, title='Top Paying Companies for
Machine Learning Engineer') + theme_minimal()

mle_count = mle_count[-46, ]

companies_ds_count %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(company, n))) +
geom_col(show.legend = FALSE, fill='rosybrown3') + labs(x=NULL, y=NULL, title='Top Paying Companies
for Data Science') + theme_minimal()


# word cloud generation

set.seed(123)
#companies_ds_count %>% with(wordcloud(company, n, max.words = 100, random.order = FALSE, colors =
brewer.pal(8, "Dark2")))

wordcloud2(head(companies_ds_count, 100), shape='circle', size = .75, backgroundColor = 'rosybrown1',
fontFamily = 'Times New Roman', color = 'random-light')
```

```r
library(tidytext)

descript = textmine %>% select(`job description`)

descript = as_tibble(descript)

descript = descript %>%  mutate(across(everything(), ~gsub("[[:punct:]]", "", .)))

data(stop_words)

text = descript %>% unnest_tokens(word, `job description`) %>% anti_join(stop_words) %>% count(word,
sort=TRUE) %>% filter(n>=100)

text %>% slice_max(order_by = n , n=25) %>% ggplot(aes(n, reorder(word, n))) + geom_col(show.legend =
FALSE, fill='wheat') + labs(y = NULL)

wordcloud2(head(text, 500), shape='square', backgroundColor='moccasin', size=.6)
```

```r
library(tm)
library(SnowballC)
#library(stopwords)

text_new = textmine %>% select(`job description`, location, `salary estimate`) %>% filter(`salary
estimate` >= 90000)

text_new = text_new %>% mutate(location = gsub(".{3}$", "", location)) # works but cuts off remote
how to fix

text_new = text_new %>% mutate(location = gsub(",", "", location))

loc_tex = text_new %>% filter(`salary estimate`>=90000) %>% select(location)

loc_tex = loc_tex %>% count(location, sort = TRUE)

loc_tex %>% slice_max(order_by = n, n=10) %>% ggplot(aes(n, reorder(location, n))) +
geom_col(show.legend =FALSE, fill='darkorange4') + labs(x=NULL, y=NULL, title='Top Paying Locations
for Data Science') + theme_minimal()

wordcloud2(head(loc_tex, 100), shape='circle', backgroundColor='moccasin', size=.8)
```

Figure 12-15: R code for Text Mining

(b) R Output

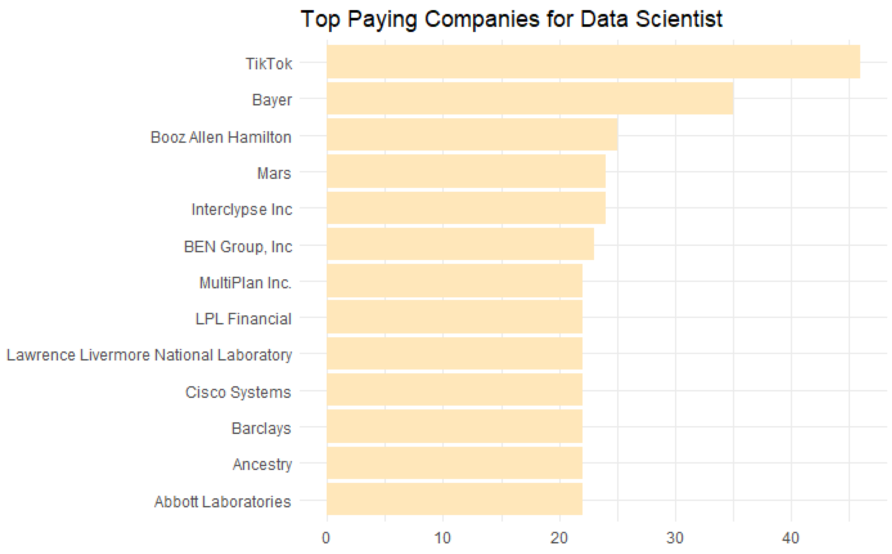


Figure 16: Bar Graph of top companies with the most jobs paying above $90,000 for a data scientist.

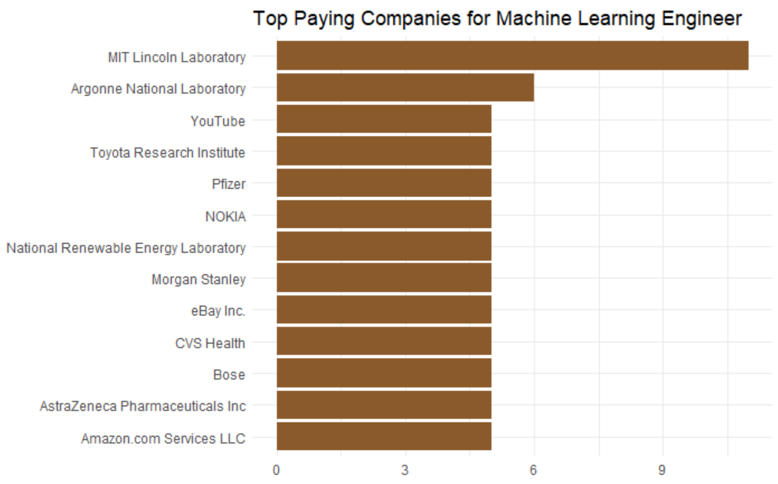


Figure 17: Bar Graph of top companies with the most jobs paying above $90,000 for a Machine Learning Engineer

.

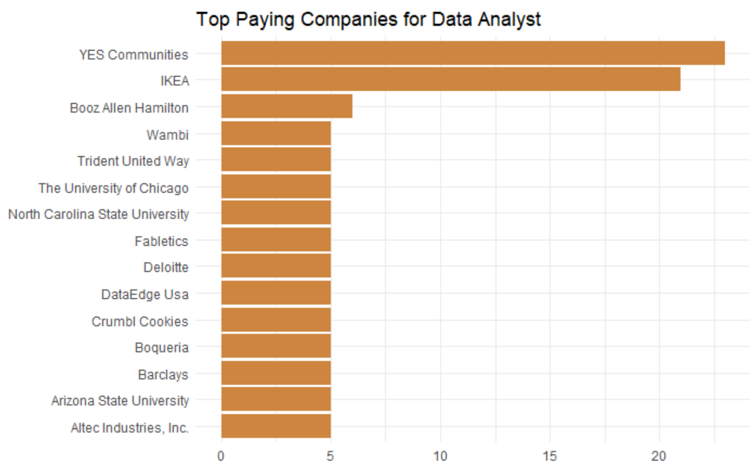


Figure 18: Bar Graph of top companies with the most jobs paying above $90,000 for a Data Analyst.

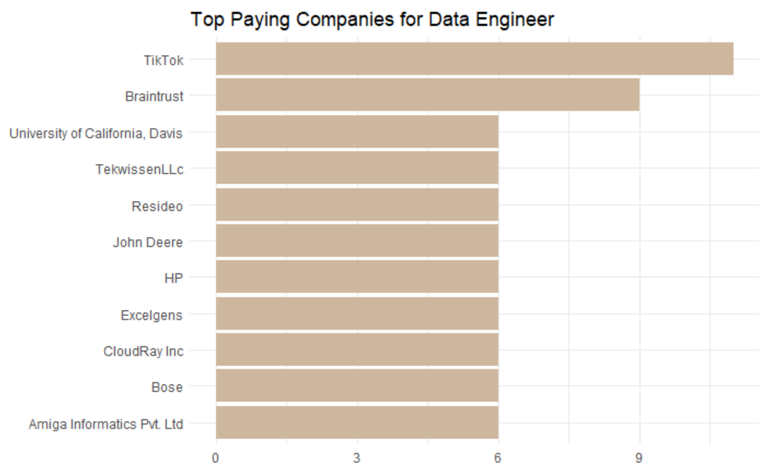


Figure 19: Bar Graph of top companies with the most jobs paying above $90,000 for a Data Engineer.

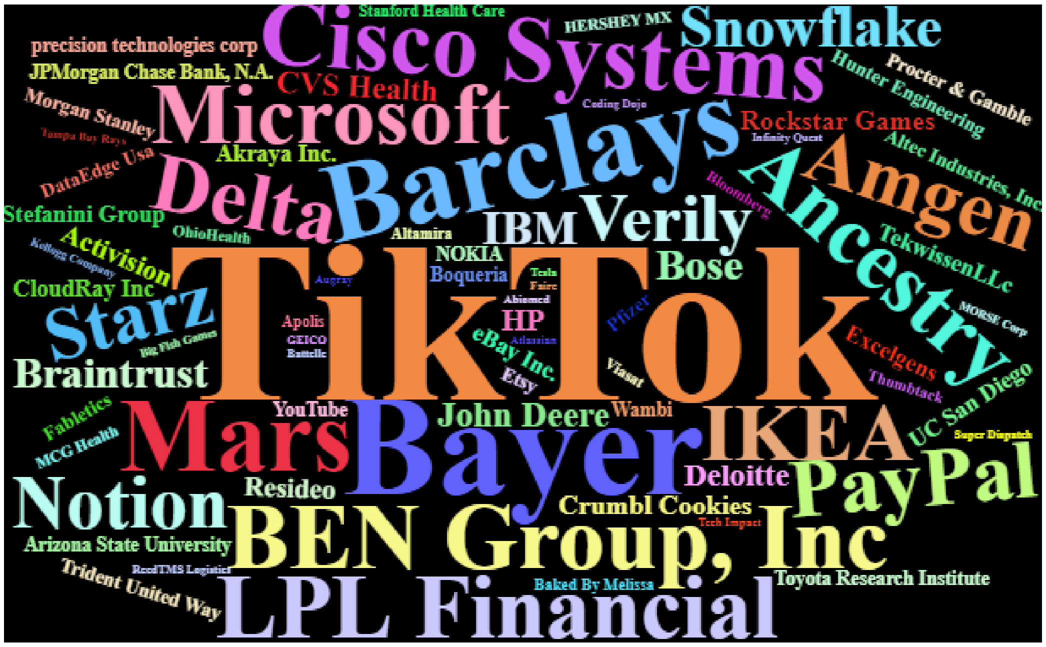


Figure 20: Word Cloud of companies with jobs paying $90,000 and above for data science positions

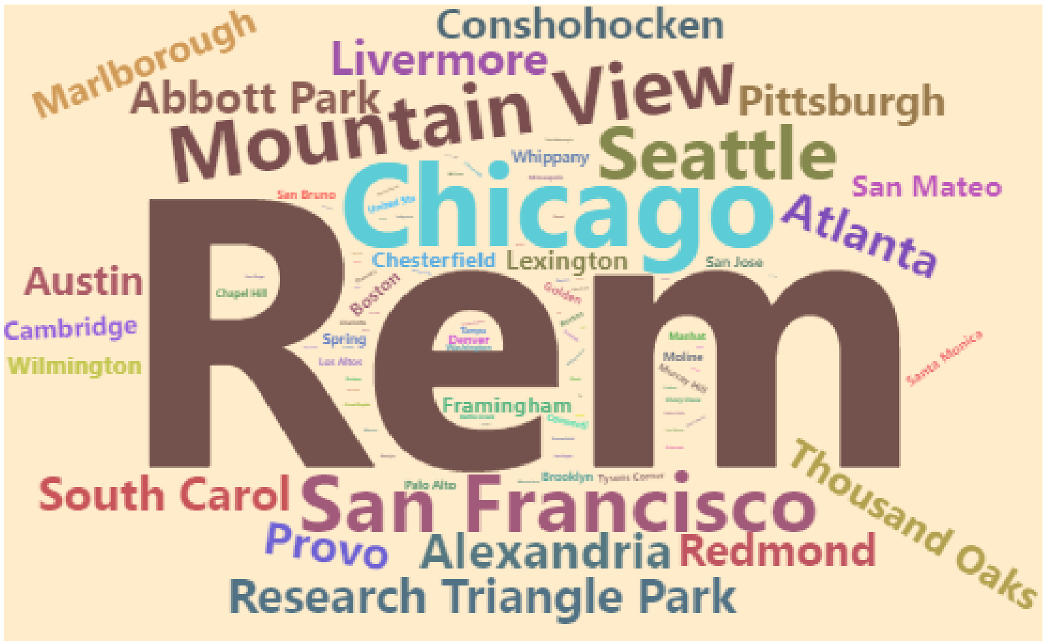


Figure 21: Word Cloud of locations with jobs paying $90,000 and above for data science positions

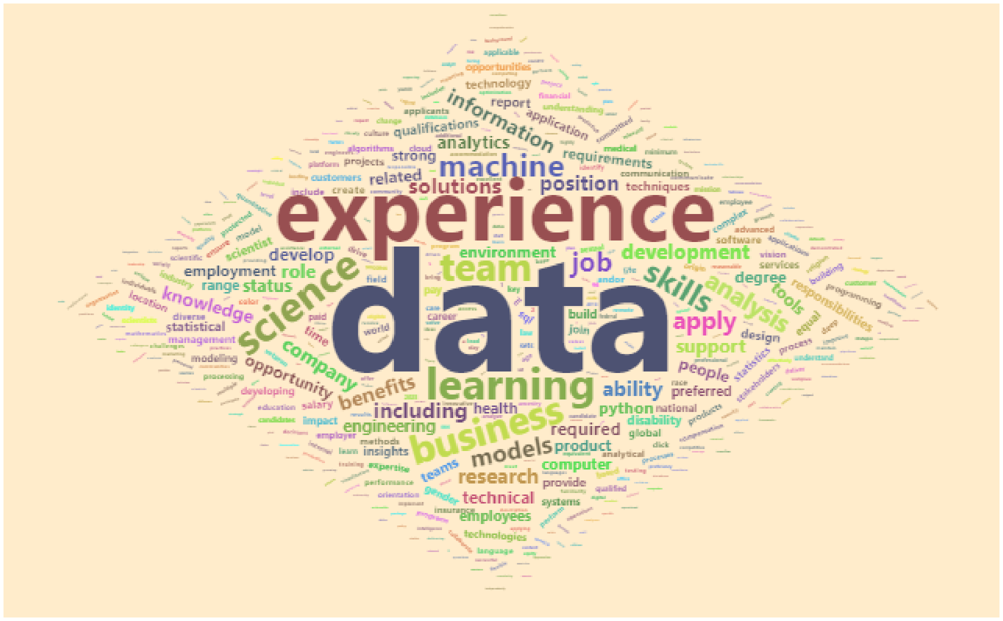


Figure 22: Word Cloud of keywords from job descriptions with jobs paying $90,000 and above for data science positions

VII. References

[1] Eddy, Nathan. “Data Scientist Salary: Skills for Unlocking Top Compensation.” *Dice Insights*, Dice, 27 Mar. 2023, https://www.dice.com/career-advice/data-scientist-salary-skills-for-unlocking-top-compensation.

[2] Galarita, Brandon. “Data Science Salary: How Much Can You Make with a Data Science Degree?” *Forbes*, Forbes Magazine, 17 Apr. 2023, https://www.forbes.com/advisor/education/data-science-salary/.

[3] Sayed, Mohamed. “Data Related Jobs in US.” *Kaggle*, 29 Jan. 2023, https://www.kaggle.com/datasets/mohamedsiika/data-related-jobs-in-us.