Report: Assignment 1 - Basic

Tanjina Islam (2609513, vu-netID:tim500), Konrad Karas (2624767, vu-netID:kks206) and Izak de Kom (2629935, vu-netID:ikm700), group 27

22 April 2018

Task 1: Explore a small dataset

Data preprocessing

We kept the following features in our preprocessed dataset:

```r
data = read.csv('ODI-2018_clean.csv')
colnames(data)
```

```
##  [1] "What.programme.are.you.in."
##  [2] "Have.you.taken.a.course.on.machine.learning."
##  [3] "Have.you.taken.a.course.on.information.retrieval."
##  [4] "Have.you.taken.a.course.on.statistics."
##  [5] "Have.you.taken.a.course.on.databases."
##  [6] "What.is.your.gender."
##  [7] "Chocolate.makes.you....."
##  [8] "birth_day"
##  [9] "birth_month"
## [10] "birth_year"
## [11] "Give.a.random.number"
## [12] "bedtime"
## [13] "good_day_1"
## [14] "good_day_2"
```

The study program feature was cleaned using the 'format_study_program.R' script. The course features were relatively clean already and did not need additional cleaning. This was also the case for the 'gender', 'chocolate makes you' and 'give a random number' features. The original 'birthday' feature was cleaned using the 'clean_birthday_bedtime.py' script and splitted into a day, month and year feature. Unfortunately, the 'bedtime' feature cleaning was problematic and was manually formatted. Finally, the 'good day' features were cleaned with the 'GoodDay_cleanup.R' script using exact string matching and the levenstein distance coefficient.

Exploratory data analysis

The cleaned data consisted of 218 samples and 14 features:

```r
nrow(data)
```

```
## [1] 218
```

```r
ncol(data)
```

```
## [1] 14
```

More men participate in the data mining techniques course:

```r
pie(table(data$What.is.your.gender.), col = rainbow(3), labels = c("Unknown: 4","Females: 63","
```

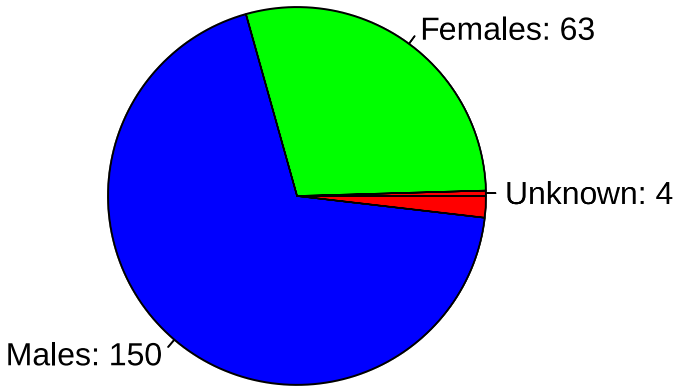


Figure 1. Pie chart showing the gender distribution in the data mining techniques course.

The birth years distribution (left out two students who were born in 1768 and 1931):

```r
plot(table(data$birth_year), xlab = "Year", ylab = "Number of students", xlim = c(1981,2000))
```

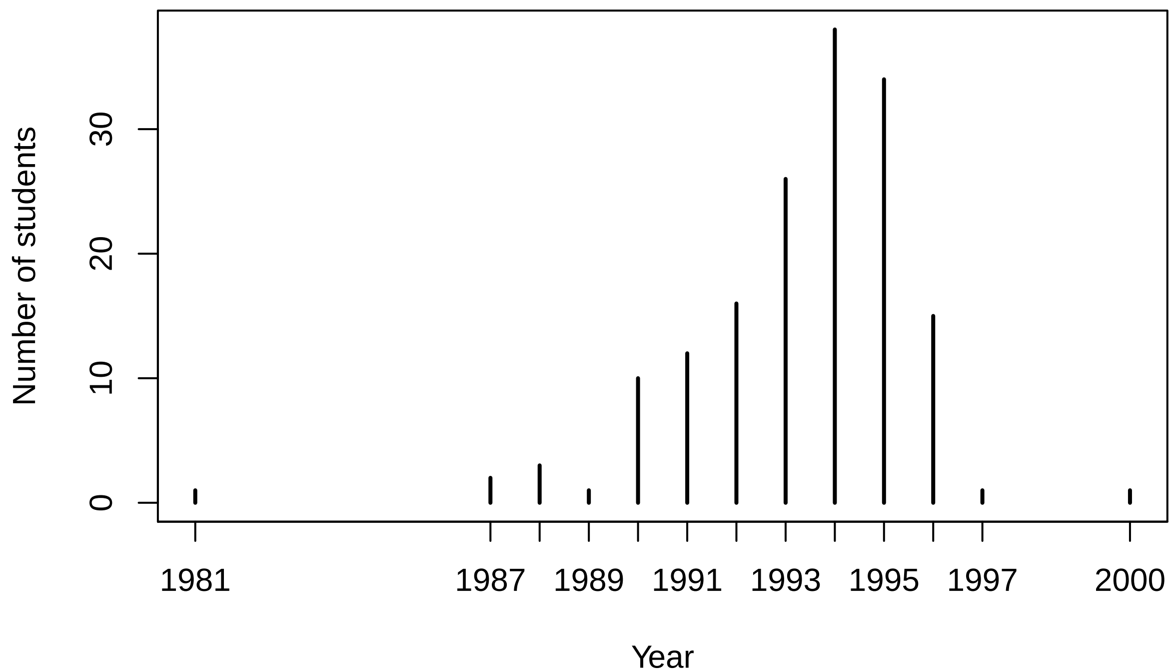


Figure 2. Birth year distribution in the data mining techniques course.

A short description of the answers of the 'Chocolate makes you. . . ' question. First, the data is converted in a frequency table using the `table` function. Then the `names` are redefined to define shorter class names. Finally, the data is plotted in figure 2.

```
choc = table(data$Chocolate.makes.you.....)
names(choc) = c("NA", "fat", "No idea", "Neither", "slim", "unknown")
barplot(choc)
```

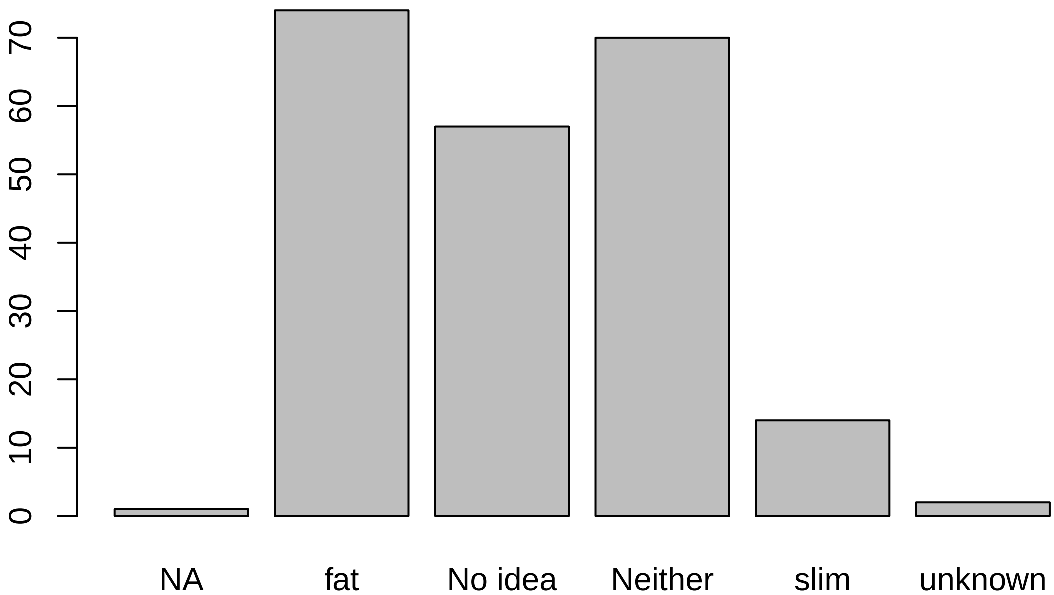


Figure 3. Chocolate makes you.... Answer frequencies.

Basic classification/regression

A simple reression using gender as dependent variable and the 'chocolate makes you...' as independent variable:

```
library(boot)
glm = glm(data$What.is.your.gender.~data$Chocolate.makes.you....., family = 'binomial')
summary(glm)
```

```
##
## Call:
## glm(formula = data$What.is.your.gender. ~ data$Chocolate.makes.you.....,
##     family = "binomial")
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.409e-06   2.409e-06   2.409e-06   2.409e-06   2.409e-06
##
## Coefficients:
##                                                                   Estimate
## (Intercept)                                                         -26.57
## data$Chocolate.makes.you.....fat                                     53.13
## data$Chocolate.makes.you.....I have no idea what you are talking about     53.13
```

```
## data$Chocolate.makes.you.....neither                                      53.13
## data$Chocolate.makes.you.....slim                                         53.13
## data$Chocolate.makes.you.....unknown                                      53.13
##                                                                       Std. Error
## (Intercept)                                                            356124.00
## data$Chocolate.makes.you.....fat                                       358522.17
## data$Chocolate.makes.you.....I have no idea what you are talking about  359234.31
## data$Chocolate.makes.you.....neither                                   358658.72
## data$Chocolate.makes.you.....slim                                      368623.31
## data$Chocolate.makes.you.....unknown                                   436160.74
##                                                                       z value
## (Intercept)                                                                 0
## data$Chocolate.makes.you.....fat                                            0
## data$Chocolate.makes.you.....I have no idea what you are talking about      0
## data$Chocolate.makes.you.....neither                                        0
## data$Chocolate.makes.you.....slim                                           0
## data$Chocolate.makes.you.....unknown                                        0
##                                                                       Pr(>|z|)
## (Intercept)                                                                  1
## data$Chocolate.makes.you.....fat                                             1
## data$Chocolate.makes.you.....I have no idea what you are talking about       1
## data$Chocolate.makes.you.....neither                                         1
## data$Chocolate.makes.you.....slim                                            1
## data$Chocolate.makes.you.....unknown                                         1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.2764e+01  on 217  degrees of freedom
## Residual deviance: 1.2647e-09  on 212  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

```r
# cross validation:
cv.glm(data[,c(6,7)],glm)
```

```
## $call
## cv.glm(data = data[, c(6, 7)], glmfit = glm)
##
## $K
## [1] 218
##
## $delta
## [1] 0.009132228 0.009132228
##
## $seed
##   [1]        403          1  766510385  508638563  812504882
##   [6] 1378222212 -433234745 2063814557 1769831188 1066600890
```

```
##  [11] -1030447099 -1803758113 -1237407642  -651520016  -573578989
##  [16]   989937329 -1688405664  2140622590    68496009 -1280293445
##  [21] -1054556278 -1032935828   102045679  1213313061   924664028
##  [26]  2049124018  -568833619   632765799  -165451122  -483167384
##  [31]  1675406187   504796425 -1175688296  1443490502   923455521
##  [36]   354194611 -1622047198  1776302100 -1295795049  -654633331
##  [41]   689600420  -899782262  1857770997  1153878095   688119638
##  [46]  1013822976  -250804413   331013857  1107377232  1624605870
##  [51]  1829347417  -394346389 -2029226534   131653276  -837488481
##  [56] -1646490891   852301708  1791637090   376146493  1234154679
##  [61]  1493035870   904973976  1137356283 -1252724839 -2103556056
##  [66]   148484310  1749448017   -37805053  1529957266  1031252260
##  [71]   417492967  1675949117  1748222644  -824815334  -766827931
##  [76]   -16185601   -38634618  1511841040  1160107443 -1884453295
##  [81]   753788416 -1151524130  -125718231  1088330971 -1018024982
##  [86]   947092364  1737374927  -767957883  1846849340     5451282
##  [91]  1910910221 -1149864377   661728302  -452906616   642771083
##  [96]  1820295785   721419832 -1657825562 -1892502911  1222852115
## [101]  1017268674 -1767087500   750879863   802704749  1873932804
## [106]  -197275350   469173909  -887984849   192010294 -1933775776
## [111]  -740326749 -1474841151 -1013808720  1579999438   694156857
## [116]   -83738549  1184297466  -896193284  1318371647 -1729315307
## [121]   722304300  1920093698  1974260957  1137430487  -319444290
## [126] -1603162888   376115931  1877985593   279743624 -1998717898
## [131]   530434161  -565282141   231268338   430961476  -590586873
## [136]  -427951651   799229652 -1195218438  2053621061 -1028845153
## [141]    14955302  1723927472  1625487443  -453872783  1861616288
## [146]    21689918   744815945    53691387  -697161782   231151916
## [151]  1832236335   -57132059   852132508  -310917390 -1032857747
## [156]  -601382745 -2080103730 -1957081048  1834576939 -1796710839
## [161] -2053007528 -1258028538   152710113  1988544883   888072674
## [166]  -236679852  -864279465  -131659699     2329572  -849024566
## [171]   479424565  -129700209   727433110  -561790144  1034415619
## [176] -1228803039  -960039152  -759337746  -698076007 -1771462357
## [181]   736385050 -1513336100   -58125601  1866878133   432122444
## [186]  -294103390  -589382147 -1347135753  -623624162  -618704168
## [191] -1602619845 -1873919015  1863987560 -1264957290  -883620335
## [196]  -571555901  -448756782 -1853194652  1012764071  1557734141
## [201]  1094096628 -1080016422 -2075065051   773653567  -983485498
## [206] -1654251440   875170931 -1062657903   277971136  -284595554
## [211]   478324329   754882203   -81462358   904749004   647402127
## [216]     8633029 -2091724164   695664338  1811536461 -1700710649
## [221]  1555684078  -725217336   863310027   186282281    43403640
## [226]   804658854   336840897  1793409363 -1217647102 -1680911044
## [231]  1684868210 -2091521936   502104100  1127992376   820162330
## [236]  -323261232  1216397116   168814484 -1708764286   849037760
## [241]   -78873028   886293904  1533099554  1328456968  1237003916
## [246]  -537092324  -210714478  2117777792 -1675598316 -1705795800
```

```
## [251]  2026183226  -203735600   381757804  1017250740   979523602
## [256]   374732256 -2128951348 -1797136160  1682216994  1265525032
## [261]  1979162124 -1652971716   505070578  2073697008  1668566596
## [266]   318751320 -1825878214 -1810897424 -2097195332 -1355237100
## [271]  1396720322 -2041023392   680264572   208934864   131555618
## [276]  -177472856  2077596556  -875644740  -445518030  2012064320
## [281]   503512244  1049436744 -1867323974 -1482653168   535622380
## [286] -1007181484  1930477842 -1044722272   361538252 -1682629024
## [291] -1880590398   780887848   560951212  -519380164  -561796750
## [296]  -635110608  1976233252 -1578521800   541892314  -687265904
## [301] -1266790660  1768270740   -38231998  1892537600   341541948
## [306] -1141325488 -1408921438  -665657016  -749398260   246006492
## [311]   714173778   948683904   457711700 -1467215896 -1590810374
## [316] -1662244016   384433964  -375838348 -1898386542  -959141792
## [321]  -940651124  -625321120   730216930 -1090817112  -770296628
## [326]   381091836   529130930  -319752720  -904033340  -657513128
## [331]    24009082 -1940716624 -1603696196 -1436600236 -1225410622
## [336]  -506712032   -96520004  1693160016 -1832078366  1548517480
## [341]   266093644  -892921604 -1425241102 -1200305920  -313835148
## [346] -1314264056  -147816518 -1186057776   297303020  -818142828
## [351]  1103434194  1892553056  -335425716 -2120197856  1849343874
## [356]  -870928664 -1125170964 -1685638084  -598525070 -1151975824
## [361]  1467061284   528239416 -2009551718 -1635052848  -904691524
## [366] -1487327596    -6801790 -1002092352  1296180028  -800531312
## [371]  -937361886  -723136504   573545740  -592040804   954531346
## [376]   287997824  1189629460  -944526808  -997100230 -1936307760
## [381]  -369675156    22358324 -1964271982   191990624   -21957812
## [386]  -661243936 -1158441054  1137018792   -79024756 -1937602372
## [391]  1968769394  -619997968 -1404424124  1961075032 -1316673990
## [396] -1197900944 -1003997124  -289906924 -2099157566 -1683784992
## [401] -1615685764 -1364402480  -996452190   103779368 -1347427060
## [406]  1842767804  -413177550  -875796032 -2091496908   806362056
## [411]  1252997306  -366171760 -1720495636   794280532  -548999022
## [416]  1761324832  -178722356  1450546656 -1462307134   -24845272
## [421]  -575819092  1100997820 -1630475790   624210224  -335244124
## [426]  -582449480  1925299546  -784042608  -548318340  1828579604
## [431]  1482163010  1649203968  -602114116  -797319344 -1432992990
## [436]  -115548856   -89298420 -1385707428   357137618  1532566016
## [441]  2087185748 -1641768472  2034928634  1954660560  1726123308
## [446]  -986471692  -919660142  -897797024  -290413300  -401927456
## [451]  2013415778  -971695448   157942348 -1199672964   102778162
## [456]   825946224 -1020943028  -273840867  -308199257 -1414516784
## [461]  1540450158  -534868453 -1700464771  -596869782   642110472
## [466]   747166897  1809316643  2015823908  1293290194  -599335129
## [471]  1808367985 -1093089354  -426656940    33408485   667828207
## [476]  1060806440 -1575311482  -486763597  -193858795  -792392334
## [481]  1774219712  -244096471  1127458459   817590124 -1429768326
## [486]  -277344081   965847993  -229589554 -1923651300 -1560224627
```

```
## [491]  2115522807  -345875456  1956273758   242467115 -1840340435
## [496]  1094591130  -659268008 -1062673983 -1407783309 -1990988908
## [501]  1079501346 -1662718665   -46429183  -675041018   -94120636
## [506]  2014836853  1700452831  -965745352   265844502  1706260259
## [511]   -67894043 -2019835646  -335498384  -319739111   -51000565
## [516] -1362225156 -1506380758   235608159  1626460137 -2081672962
## [521] -1731436308   468845629  -886474745  -160398352  -177975410
## [526] -1212723781  -272684131   419934538  1716246888  1404358481
## [531]   517786691  -697330364 -1097681422   130072519  -416480879
## [536]  1928770838   756960628 -2006430715  1859533903 -1510911928
## [541] -1201093146   396368659   -92570507 -1574705582 -1732665184
## [546]  1345485193 -1471481413  1848432140  -756756582 -1530300657
## [551] -2093290151   588318702   883486332  -319367443   -99442281
## [556]  1996024416  -599744450  -293803765 -1445270643    18229434
## [561]   -12194056   904766625   -89391021   500706164  1353636098
## [566]   543586711  1884152673 -1527236698   -63925852   764299349
## [571] -2059346689  1234264472 -1299547466   115184451 -1204744571
## [576]  -524106974   -84477936  -516799559  1676551275   495875932
## [581]  -641446262 -1036608897  -915012663  1102159966 -1876540660
## [586]  1070754525 -1189444633 -1261283312   854209838   892340699
## [591] -1270624579    23870506 -2040054584   829413873  -884828317
## [596]  -620310684 -1389844846  1465746279  -511365839  -744978826
## [601] -1804634220  -614923355   347059375  -582686360  -846327482
## [606]  1291348211 -2104592811  1960188466  1370957824  1407040745
## [611]  1909537627  1818451372 -1515600454  1579582319   740709625
## [616]  1547618702 -1572979620  -612528435    -7132617   998596800
## [621]   674066334   141309035  -723020819  1630385754 -1354991336
## [626]  1295112069
```

There seems to be no clear relationship between the answer type and the gender.

Task 2: Kaggle Competition: Titanic: Machine Learning from Disaster

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

The aim of the competition is to predict who among the passengers and crew was more likely to survive than others. Kaggle provides two datasets: *train* and *test*. While both datasets reference to passengers details, only *train* dataset contains infromation if passenger survived or not. Our goal is to predict which passenger from *test* dataset survived the sinking of Titanic.

Preparation

As mentioned before, Kaggle has provided two seperate datasets: *train* and *test*. Both datasets contain details about passengers and their trip. The only defference between them is *Survived* column in *train* dataset that indicates if passenger has survived the Disaster. That dataset will be used to train our models.

Before attempting to perform predicions, we focused on given data and tried to retrieve more interesting facts based on Feature Engineering. Because the only column that differ is *Survival*, for further processing we decided to datane both datasets into big one. Such an approach allows us to perform more adequate data analyse as we have a full insight of traveling passengers.

Data exploration

To have a better insight into assignment, we had to explore given data. That's the most important step - we have to be aware of all the details to work efficiently. Whole dataned dataset contains 1309 records (passengers) with 12 variables. In this part we will take a closer look to every attribute.

In datasets we can distinguish several (12) columns:

- Survived - indicated if given passenged survived
- PassengerId - passenger index in dataset
- Pclass - the ticket class (1,2,3)
- Name - full name of passenger, including their title
- Sex - sex of passenger
- Age - age of passenger
- SibSp - number of siblings or spouses traveling with passenger
- Parch - number of parents or childern traveling with passenger
- Ticket - ticket number
- Fare - passenger fare
- Cabin - passenger's cabin number
- Embarked - port of embarkation (C = Cherbourg, Q - Queenstown, S = Southampton)

Lets make a closer look into several variables.

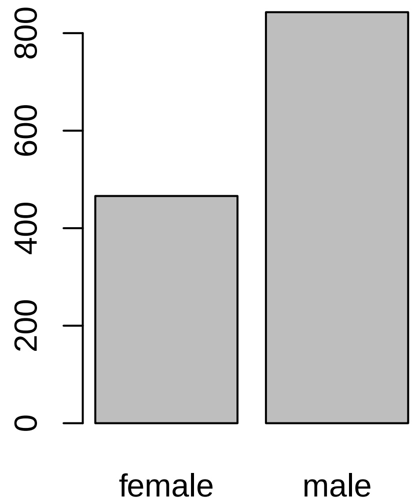
Name

In given dataset we can see that *Name* attribute contains string with passenger's name, surname and title.

example: *Allison, Master. Hudson Trevor*

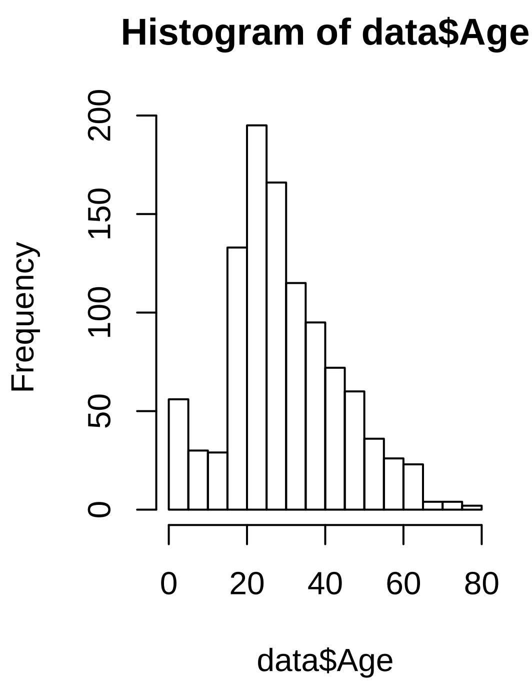
Fortunately, all rows in *Name* column follow the same string pattern (*surname*, *title first name*). Thanks to this fact, we will be able to retrieve more additional infrmation about passengers, like common surnames or titles.

Sex



Investigating Sex attribute we can see that there were 466 females and 843 males onboard. That gives us the first easy grouping of passengers.

Age



Regarding Age attribute, we can see that this variable varies up to 80 with mean around 23

Feature Engineering

After investigation of given dataset we can distingush columns that seem to be useful for further processing to retrieve even more data. In such an approach we are able to create additional columns with relevant variables that sould result in better prediciton accuracy.

Feature: Title

As mentioned before, Name column contains not only name and surname of passenger but also a title (like Sir., Mr., Mrs., …). Following common pattern (*surname*, *title first name*) we can retrieve additional Column in our dataset that would group our passenger by Title. In addition, groups of unique similar titles were replaced by the tame variable (like 'Capt', 'Don', 'Major', 'Sir' => 'Sir').

As a result we obtained a fixed set of values with 11 levels.

Feature: Family

Basing on variables *SibSp* (number of siblings or spouses), *Parch*(number of parents or child) and Surnames retrieved from *Name* variable we are able to group passengers by families. Assuming that during disaster, every person takes care about their relatives, we think that it can be a significant factor in predicitons.

Our assumptions:

- the number of relatives with who each passenger was traveling is aclculated as follows: *SibSp + Parch + 1* - result is family size
- if family size is less or equal 2 we assume that the value is not revelant and we mark sucha afamily as *n/a*

As a result we obtained Family attribute with 97 levels.

Feature: Deck

Analysing *Cabin* attribute we figured out that each cabin number consists of Deck Level and Room number (like C40 => Deck C, Room 40). Because Deck Level could play important role in evacuation, we assumed thet it's a siginicant attribute. We decided to create a new attribute called Deck and we assigned relevant Deck Level to each passenger. Unfortunately, not every passenger had a Cabin number assigned, in such a case we marked Deck as 'U'.

Result:

```
##    A    B    C    D    E    F    G    T    U
##   22   65   94   46   41   21    5    1 1014
```

Feature: TicketType

Looking into ticket numbers we can see that some tickets have common prefix that could refer to Ticket Type of place of purchase (example: STON/02 42342). We decided to retrieve that ticket prefix and create a new attribute for each passenger. If ticket didn't have any prefix, we marked TicketType as 'num'.

As a result we obtained TicketType factor with 51 levels.

Missing values

We have found that some records lack in Age attribute. In such a situationw e decided to use a Decision tree to predict missing Age values. As siginicant factors we marked attributes: Pclass, Sex, FamilySize, Embarked, Title, SibSp, Parch.

Also Fare column had some missing values. In such a case we replaces missing values with median of all ticket Fares.

Classification and evaluation

By analysing our data and engineering some additional features we have enriched our dataset.

Within all calumns we decided that only few of them play siginificant role in predictions.

Chosen factors: *Pclass, TicketType, Sex, Deck, Age, SibSp, Parch, Fare, Embarked, Title, FamilySize, FamilyID*

Creating a steup

To evaluate classifiers we will need to create a proper setup. In this case we decided to use *train* data from Kaggle as it contains *Survived* column. For evaluation purposes we decided to split the data for training and testing sets with ratio 70/30 (70% - training, 30% - testing). While splitting the data we based on random ordering.

For evaluation we decided to use two non-linear algorithms: k-Nearest Neighbour Classification and Conditional inference trees. Both classifiers were trained and tested with the same sets of data. For evaluation analysis we used Confusion Matrix.

Factors that we took into account:

- Accuracy - how well results were predicted
- 95 CI - confidence intervals, our final score should match into calculated intervals
- Kappa - accuracy through random predicitons
- F1 - model that takes recall and precision into account

Evaluation of k-Nearest Neighbour Classification

Accuracy : 0.6929
95% CI : (0.6338, 0.7477)
Kappa : 0.3338
F1 : 0.5638

Evaluation of Conditional inference trees

Accuracy : 0.809
95% CI : (0.7566, 0.8543)

Kappa : 0.5929
F1 : 0.7437

Kaggle Submission

For Kaggle competition we decided to use Conditional inference trees as it gives us higher results in included evaluation factors.

We have submited our Prediction in Kaggle system and obtained satisfactory result 0.82296 which is top 3% in leaderboard (username VUDM27). This result also matches into expected Confidence Intervals calculated during evaluation.

Task 3: Research and theory

Task 3.A – Research: State of the art solutions (10 points)

Data mining competition: "Planet: Understanding the Amazon from Space", July 2017

In this data mining competition, the participants were provided with a dataset consisting of over 40,000 satelite images. Each image had a resolution of 256 by 256 pixels, and covered an area of approximately 950 by 950 meters. The satelite images were captured above the Amazon in South-America. The goal of the competition was to successfully classify these satelite images. Each image could be classified into multiple classes:

- **Atmospheric conditions**: clear, partly cloudy, cloudy, and haze
- **Common land cover and land use types**: rainforest, agriculture, rivers, towns/cities, roads, cultivation, and bare ground
- **Rare land cover and land use types**: slash and burn, selective logging, blooming, conventional mining, artisanal mining, and blow down

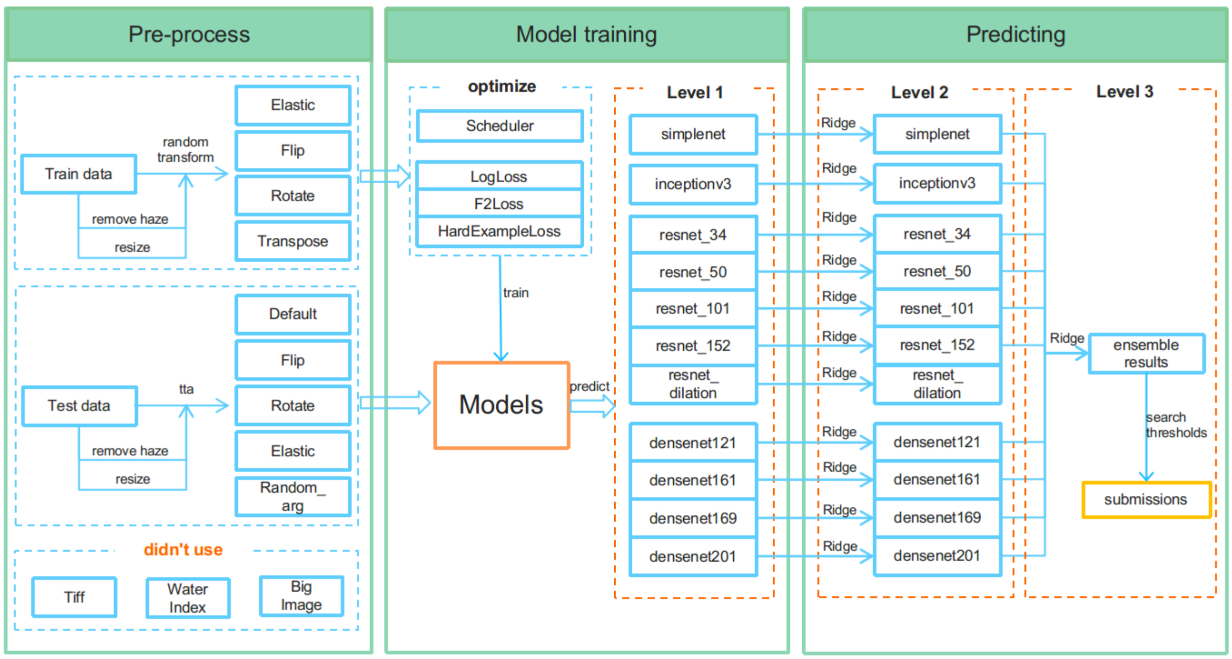
The participants' model predictions were evaluated with their mean F2 score. This score determines the model prediction accuracy using the precision and recall measures. The formula of the score:

$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \quad \text{where } p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}, \ \beta = 2.$$

The winner of the competition was a Kaggle user named "bestfitting". He managed to win the competition by fine-tuning 11 convolutional neural networks (CNN's), and using these CNN's to build an ensemble model. This ensemble model was used to predict the final classes.

The pre-processing step consisted of resizing the satelite images, removing haze from the images, and augmenting the data (e.g. flipping, rotating and transposing). Next, one *simplenet*, one *inceptionv3*, five *resnet* and four *densenet* CNN's were trained on the labeled training data. The resulting models were then ensembled by using a ridge regression model, allowing for the selection of the strongest models for each label prediciton.

The main reasons why this model won the competition was the creation of an ensemble model. As different models had different capabilities on each class label, the combination of the models resulted in a higher accuracy.



Task 3.B – Theory: MSE verse MAE

Mean Squared Error(MSE)

The mean squared error (MSE) of an estimator measures the average of the squares of the errors that is, the difference between the actuals and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where, $\hat{Y}_i$ is a vector of n predictions and Y is the vector of observed values of the variable being predicted.

Mean Absolute Error(MAE)

The mean absolute error(MAE) measures the mean of the absolute errors that is, the absolute value of the difference between the forecasted value and the actual value. MAE tells us how big of an error we can expect from the forecast on average.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y_i}|$$

Where, $\hat{Y_i}$ is a vector of n forcasts and Y is the vector of actual values of the variable being predicted.

MSE Vs MAE

Mean squared error has the disadvantage of heavily weighting outliers. It is a result of the squaring of each term, which effectively weights large errors more heavily than small ones. Where this kind of property is undesirable, MAE can be used in those applications by the researcher.

When dealing with outliers, it might be helpful to use MAE instead of MSE since MSE gives higher error than MAE. Yet, MSE is more popular and efficient than MAE, because MSE punishes larger errors, which tends to be useful in the real world.

The mean absolute error (MAE) has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units.

Both MSE and MAE are scale-dependent.For instance, if the observed data are in km then MSE is in km^2 and MAE is always in km respectively. Often, we need to perform accuray test on predicted values across different units. In that particular context, both MSE and MAE will not be applicable because they can only be compared between models whose errors are measured in the same units.

For evenly distributed errors that is, when all of the errors have the same magnitude, then Root mean squared error(RMSE) and Mean absolute error(MAE) will give the same result.If the square of the difference between actual values and forcasted values gives a positive distance which is same as their absolute distance then, MSE = MAE.

Data collection and exploration

To calculate MSE and MAE of different regression methods we used the *Energy_efficiency.csv* dataset.This dataset has been collected from the UCI MAchine Learning *Repository*[3]. This dataset is a collection of 768 samples and 8 features, aiming to predict two real valued responses.

The dataset contains the following eight attributes or features(X1,X2,. . . .X8) along with two response variables(Y1,Y2):

- Relative Compactness(X1)
- Surface Area(X2)
- Wall Area(X3)
- Roof Area(X4)
- Overall Height(X5)
- Orientation(X6)
- Glazing Area(X7)
- Glazing Area Distribution(X8)
- Heating Load(Y1)
- Cooling Load(Y2)

It is important to implement energy efficiency in building to mitigate the impact of climate change. Due to the high demand for energy and unsustainable supplies, energy efficiency in building plays a vital role reducing energy costs and greenhouse gas emissions. Therefore, studying this dataset to evaluate how well energy is being used there to cut out the costs which will be helpful to have a ECO-friendly environment.

Experiment and perform evaluation

We load the samples into a dataframe and took all the column attributes as factor. We randomize the data frame using . Then, we divided the dataset into a trained dataset with the top 80% of the samples, and a tested dataset with the bottom 20% of the samples respectively. So, energy train data has first *614* entries from the dataset and energy test data contains the rest *154* samples.

At first we set up a model*(rt1)* for tree regression using the *Heating.Load* as outcome variable and all the eight attribtes as input variables and fit a new dataframe with the actual and predicted value of the model based on the test data. Using Regression Tree model(rt1) and "Heating.Load" as outcome, we calculated MSE = 6.59 and MAE = 2.101.

Similarly we fit another model(rt2) for tree regression but instead of using *Heating.Load* as outcome variable now we are intersted to use *Cooling.Load* as outcome variable. And we figured out for this model(rt2), using *Cooling.Load* we got MSE = 8.461 and MAE = 2.084.

We randomize the data frame using again. Next up we fit two models namely rf1 and rf2 respectively for both *Heating.Load* and *Cooling.Load* as outcome variables using Random forest regression following the same approach as described earlier for rt1 and rt2. Then we measured the MSE and MAE and for rf1 we got, MSE = 1.36 and MAE = 0.907.

```
##                           % Inc MSE
## Glazing.Area                   74.7
## Glazing.Area.Distribution      41.0
## Relative.Compactness           24.7
## Surface.Area                   24.1
## Wall.Area                      20.8
## Roof.Area                      18.9
## Overall.Height                 17.7
## Orientation                   -19.6
```

Observing the result of *importance()* function to calculate the importance of each variable, we got to see that *Glazing.Area* was considered the most important predictor; it is estimated that, in the absence of that variable, the error would increase by 74.7%.

Whereas for model rf2, using *Cooling.Load* we got MSE = 3.698 and MAE = 1.348.

```
##                           % Inc MSE
## Glazing.Area                  75.51
## Glazing.Area.Distribution     29.36
## Relative.Compactness          24.03
## Surface.Area                  22.02
## Roof.Area                     21.45
## Wall.Area                     20.14
```

```
## Overall.Height                   18.78
## Orientation                      -4.36
```

If we look into the *importance()* function to calculate the importance of each variable, we can see that The *Glazing.Area* was considered the most important predictor for *rf2*. it is estimated that, in the absence of that variable, the error would increase by 75.51%.

If we perform overall evaluation and compare MSE and MAE for all four models we can see that using random forest regression the model, rf1 with *Heating.Load* as response variable has lower error rate for both MSE = 1.36 and MAE = 0.907 compared to other models. For regression tree model both rt1 and rt2 produced relatively higher MSE values though MAE values did not varry significantly.

Task 3.C – Theory: Analyze a less obvious dataset

Theory: Analyze a less obvious dataset : SMS Spam Filtering

Text message classification requires supervised Natural language processing techniques to filter messages with respect to it's types and maps inputs to it's targeted varaibles based on the learning information which it gets from trained data.

Our aim is to predict the probabilities of a message being spam or ham. Therefore, we need to perform text mining on unstructured data, fit a predictive model on top of that and suggest improvement if any to increase our proposed model's performance.

Data Colection

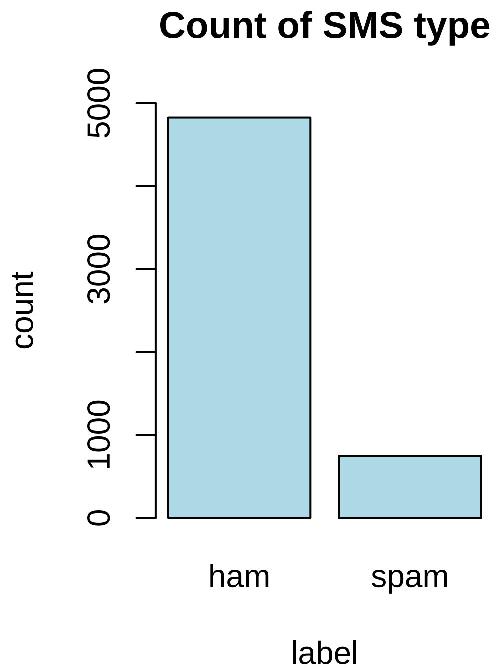
The dataset: *SmsCollection.csv* has been collected from the course website. This dataset is a collection of 5574 text messages in English provided by a UK forum for research purpose. In this dataset, messages are labeled as either *spam* or *ham*. *Ham* stands for legitimate message whereas the type *spam* is used for trashed or unwanted message.

At first we load the data from the source. Then we split label and text and bind them into a dataframe.

Data exploration

The *SmsCollection* dataset contains text messages only. Since we are only dealing with text messages which are unstructured in nature, so we will need to perform some basic natural language processing technique in order to tokenize those texts, computing the frequencies of words, calculating document-feature matrix and so on.

In general, almost all the classifiers use a conditional probability model to classify data. Looking at the samples we can see that they are mainly concerning about classifying the messages into a two class problem as spam or ham. Among 5574 text messages there are 4827 messages categorized as ham and the rest 747 messages are classified as spam. We generate a barplot of it.



As we can observe there are more ham messages than spam There are various classifier algorithms to solve this but we found Naive Bayes as the most suitable one for this purpose. Naive Bayes is a simple yet powerful classifier based on Bayes probability theorem which uses conditional probabilty model. It is more suited to categorial variables although it can also be used for continuous variables. Text messages are often noisy and the amount of predictors are way more than the actual samples. Naive Bayes classifier follows conditional independence theorem. Therefore, it assumes that features are independent of one another which is a high bias and this introduced strong bias might be helpful in reducing the variance to achieve better predictions.

Data Processing and transformation

We load the samples into a dataframe and use the *label* as a factor while on the otherhand we are using attribute *text* as character.And then we randomize the data frame using . To process the text data we transformed the data frame into a volatile corpus as they cannot be directly handled by a data frame. VCorpus converted each of the messages as a document.

In the VCorpus text document each SMS has it's content in raw formatted way. So, before applying Naive Bayes classification algorithm we need to clean up data. It will help the algorithm to perform more efficiently which will eventually increase the accuracy of predictions.

Our data cleaning process includes : conversion of all texts to lowercase, removal of numbers that is neither a spam nor a ham, removal of some common stop words in english such as: "a","an","the","for" etc. that neither indicate spam or ham, punctuation and extra whitespace removal. Finally after completing data cleaning task, final version of the VCorpus were transformed into a Document-Term-Matrix (DTM) that will be taken into account as the basis for the classification.Doing so, we found 7713 unique terms in total for all 5574 entries.

Generating training and testing dataset

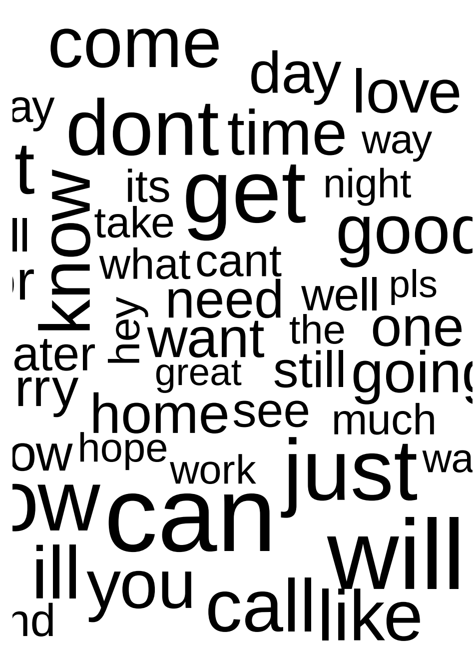
We divided the DTM to generate our training and testing dataset. The Document-term-matrix is splited into a trained dataset with the top 75% of the raw sms data, and a tested dataset with the bottom 25% of the raw sms data using the *createDataPartition()* function. Since, we only need "label" attribute of the raw sms dataset we created two classifier labels namely "sms train labels"" and "sms test labels" by splitting with exact same proportions of row that we used before. We made these two classifier labels to use them for Naive Bayes model later on.

Table 1: Frequency comparison among different datasets based on SMS label

	Raw Dataset	Training Dataset	Test Dataset
ham	86.6	86.6	86.6
spam	13.4	13.4	13.4

Using prop.table() we converted number of spam/ham messages of both sms train and test labels into fractional values and preserved those proportions into our train and test dataset. Looking into the above table we can see that 86.6% of the messages correspond to legitimate messages (ham) and 13.4% to spam messages which follows the same proportion in each of our dataset perfectly.

We created a wordcloud from the cleaned vcorpus to look at the most frequent words in the available bag of words. We also created seperate wordclouds for spam and ham messages where most frequent words appeared in larger font and less frequent words in smaller font.



Looking the above wordclouds we found that the spam contains "call", "now", "free", "mobile" as most frequent words whereas ham contains frequent words such as "will", "get", "now","just", "can". Also spam(on left) showed extreme frequency in it's wordcloud. Since, it seemed that our datasets contains distintive words, we hope our choosen classifier algorithm(Naive Bayes) will be a good fit for sms prediction.

We removed most of the least frequent words from the DTM and created final train and test dataset that we should be using for the training using only the most frequent words that appeared at least 5 times in datasets. The number of columns for each trained and tested datasets are then shrink from 7713 terms to 1193 column (words).

Training the data using Naive Bayes Model

we already have trained and tested labels respective to the datasets.And we used naive_bayes() to train and build model based on the trained dataset along with it's trained label. Our trained model contained information from both trained and tested DTM whcih have 1193 distict words(possibilities of either spam/ham).

Evaluate performance

evaluating it's performance we can see that Naive Bayes has accuracy rate 96.77% with sensitivity 78.49% and there are 5 *spam* text messages wrongly classified as ham and 40 *ham* text examples wrongly classified as spam. In order to improve it's performance we used Laplace along with it and laplace lowered both of the false positive and false negative values and increased our accuracy up to 97.56%. The summarised version of their performances are given below into tabular form.

Table 2: Performance Table for two models

	True_Neg	True_Pos	False_Neg	False_Pos	accuracy	sensitivity
Model-1:[**Naive Bayes**]	1201	151	35	5	0.9713	0.8118
Model-2:[**Naive Bayes+laplace**]	1204	163	23	2	0.982	0.8763

Conclusion

To solve this task we classified text messages as ham or spam using some basic natural language processing and then model a naive Bayes text classifier. There are numerous ways of doing this but using the Naived Bayes classfication algorithm, we obtained more than 97% accuracy in predicting whether a new incoming message is a spam or not based on it's training data.

References

1. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
2. https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
3. https://www.r-bloggers.com/understanding-naive-bayes-classifier-using-r/
4. https://datascienceplus.com/text-message-classification/
5. https://en.wikipedia.org/wiki/Mean_squared_error
6. https://en.wikipedia.org/wiki/Mean_absolute_error
7. http://archive.ics.uci.edu/ml/datasets/Energy+efficiency#

8. A. Tsanas, A. Xifara: ‘Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools’, Energy and Buildings, Vol. 49, pp. 560-567, 2012 (the paper can be accessed from weblink)