

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:

```
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.

More men participate in the data mining techniques course:

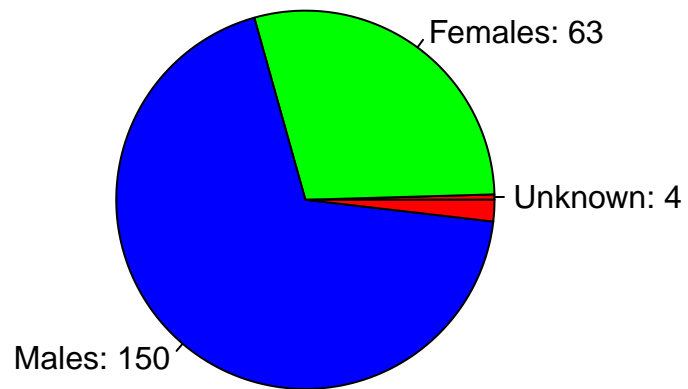


Figure 1. Pie chart showing the gender distribution in the data mining techniques course.

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

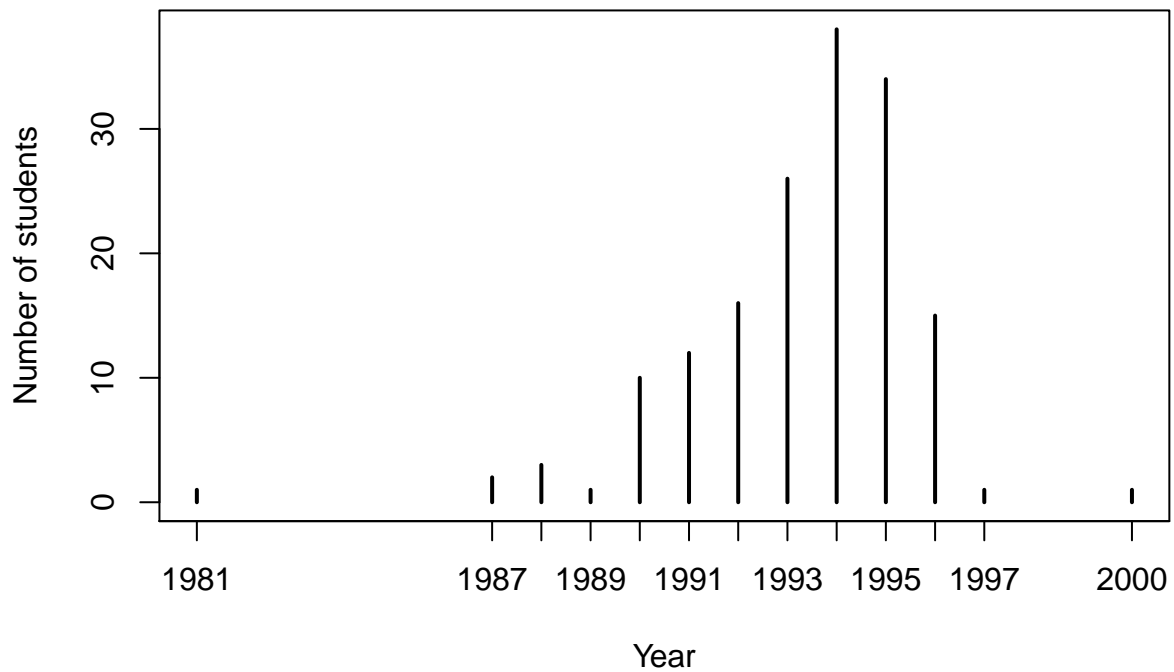


Figure 2. Birth year distribution in the data mining techniques course.

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.

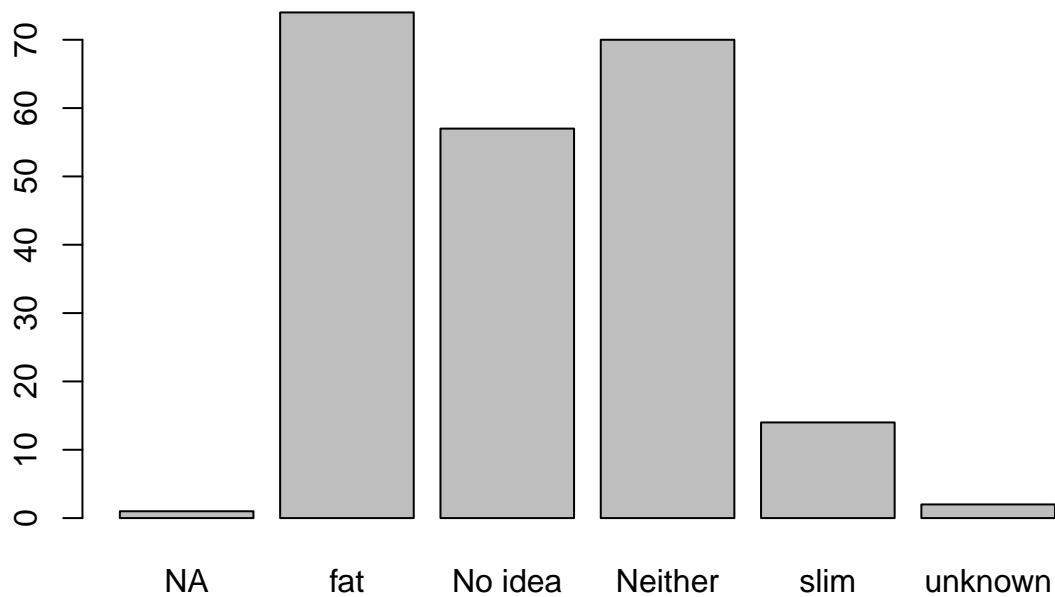


Figure 3. Chocolate makes you... Answer frequencies.

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
# 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 -2076958581 -1278641559 -2128165320
## [6] 479887078 -921544063 2144077331 -953871934 -100935052
## [11] -430441353 -2090498707 2110724612 970635562 50040469
## [16] -1537383121 323919926 1422595168 459184803 -672191039
## [21] -933279312 -1926457138 -1932761543 -1999390645 -80991750
## [26] -1343822596 -914175681 -1709177323 1882559788 970479106
## [31] -331453219 169514967 -961224002 -259099400 822741723
## [36] 1487147321 267768968 1336775734 1759670385 -640490845
## [41] -1192658958 -1001678012 199504903 1374999005 -1165914412
## [46] -509562374 518806853 -1164105313 1875794726 -834796624
## [51] 656150611 -2082059407 -1341261152 1237192254 246048073
## [56] -26272773 -119447606 1904095532 883273007 1406194661
## [61] 1514621596 -586101006 -624083091 903780519 -169837874
## [66] 1017243688 1888121899 -950110647 -1393328296 437238278
## [71] -2007931935 385286515 -685690398 1983415636 -738706345
## [76] -1790328755 -801553436 -2117099062 -439883723 -892393841
## [81] 194566038 -451855552 1431238147 1884326945 1679662864
## [86] 1480307950 -742499175 1442822443 1604710426 1247275740
## [91] 1536555743 -700382027 -1389465012 674021026 1771308541
## [96] -623120649 -412350434 -2074875176 -44032453 595069913
## [101] 275683688 550088854 -145191407 -1527601213 2011749330
## [106] -143317404 139195815 509520637 1129066228 -672511526
## [111] -29192923 -1955177409 1173425094 -2132318128 158139507
## [116] -433493871 1884764352 -2008272226 -388632983 2113715867
## [121] 1141789610 1452160972 120359567 -1536286011 -1785329028
## [126] -1356626222 1281077837 1601888007 117211886 -1273366584
## [131] 1431894219 -535279319 1201549688 -380389722 -1530707775
## [136] -361526957 1913714178 231132980 1165499575 -1928319827
## [141] 636639684 528284138 441189973 -466200593 1170008054
## [146] -920051424 -906899741 -42371199 -398315024 1276293262
## [151] -2098055175 1628097931 -375998022 -1601603396 1597709311
## [156] -1542780587 -1713581460 1731458754 1435762973 -1474300009
## [161] 660796158 -325859144 -1697721701 1931075833 -852983224
## [166] 1338084726 -946422607 -1879363357 -115568718 -355430396
## [171] 1898726215 -1147561443 -596774764 -1400717254 -1247092091
## [176] 1434100575 -20079642 -2037639568 1564711315 -1145217487
## [181] 1402033376 -1706867330 1417893897 -632399813 -703617526
## [186] 2037994988 2031475311 -350431835 -2063432356 1894717746
## [191] -774428627 -807180569 434473998 1325700584 2044813291
## [196] 1666719881 -441895912 1964475206 -603094623 -235974605
## [201] 743480738 -878843756 -337141993 -605595379 372256804
## [206] 1094504458 -32819851 243446991 1281275862 -897157248
## [211] -1748259901 -224985759 -1713855536 1349112622 670904025
## [216] 1072831723 -2035786406 680603420 1630439967 764631669
## [221] -495905012 312543970 430108605 -2094884809 641733342
## [226] 498470424 -1656417925 1507961113 461403560 449526624
## [231] -739034654 1603228584 1822758092 1431097340 877389234

```

## [236]	1479333360	-286004284	849016152	-925755014	-171211856
## [241]	1469684156	-1894938540	-1861711934	-8400864	1511524540
## [246]	-164142512	-1118657566	963797096	-1429932980	988260604
## [251]	-2109838350	1171220736	-1935650444	483736584	-555724870
## [256]	-1302674992	624077804	1614197140	241463762	787292512
## [261]	-459284660	-1953490656	1232990082	-1966727448	338759916
## [266]	-1576684484	-1913869454	1217429104	-1416416220	-769602760
## [271]	41815194	859828944	-704380740	-624136556	-776337790
## [276]	-870000448	-657951940	-376464240	-2036376030	878379016
## [281]	1367293196	-2111611748	-1412047342	1232060288	1732619796
## [286]	-1717482968	903873850	-1515480624	-57314196	1592523060
## [291]	1159636626	665275232	1695482700	-1209509920	-793106526
## [296]	2115127208	1947487628	517531836	1626589554	164934896
## [301]	1098219588	1007571288	-865962438	1603132272	-1429456836
## [306]	107388692	-1412123198	-1370703136	-1260804228	-979996976
## [311]	1704310946	-636642264	1542055180	96528316	971139378
## [316]	-517690944	1495108148	-654431288	-652871494	667900304
## [321]	31420908	-1181924780	-21938030	2093580064	-1000150580
## [326]	1125373408	1029519042	-866032600	329282220	1654080700
## [331]	-208274958	-178290384	-1767631708	-1214634312	1471712602
## [336]	-1631804016	-966683780	653572372	-599174334	-1136887040
## [341]	-1982662724	-1466319024	-599489758	-1544594104	-1752434164
## [346]	291526748	1784564946	-1156294144	1062260052	-893101592
## [351]	-1542022150	936194256	-1522352852	-1363807500	2112940434
## [356]	-1141083040	-124123892	-1483664672	-1771009694	-1117369688
## [361]	-1537060276	62243196	-1904114382	-2016571280	337114948
## [366]	-660124200	-1273619334	-1749160400	353733308	-404353196
## [371]	-1213467582	-963867232	-1275045188	31693264	-2058990238
## [376]	1254633448	895428556	820575996	115139058	1134741632
## [381]	-1305356	-255003000	-1142749382	1663718224	1271527020
## [386]	-1166684012	-600497838	-141508896	-721390900	368988320
## [391]	727923586	1730294504	2095174380	738159676	-1630805390
## [396]	1900987504	-714504668	-1420647368	-464631782	48351952
## [401]	-1505057732	1596994708	926542978	219613120	-451135172
## [406]	-1201433968	-2118616030	-1871469560	-1454573428	-474379236
## [411]	-1475533166	1663243392	1446043668	-232708824	-1829622214
## [416]	516924624	1627718764	294310580	-407115758	-573777952
## [421]	2001260492	-1355420192	1098886690	372730152	1207349516
## [426]	1697134140	-1522474766	-667714832	-1570167996	568322648
## [431]	-968863174	-1967896848	484998332	2076839444	1539219650
## [436]	-142888864	-489435780	1826790608	-501392350	949197480
## [441]	366210956	-114468420	763687986	608474944	577173684
## [446]	-835386552	759880378	1107037968	503093484	-523439020
## [451]	-683817198	-1420708960	841998284	2078632800	249262274
## [456]	-1283238360	1793812378	1097248527	1920333145	-2037428242
## [461]	834442876	-809468179	798165399	1220600416	371243070
## [466]	1029603595	-1319603315	-1820962630	-496265480	1627487393
## [471]	-1914607533	-1812319372	-909217534	1469821335	277744481

```
## [476] 1750495142 591165860 725008469 1228731647 162699672
## [481] 1297441462 1622244163 -523139451 -817468638 475959312
## [486] -1644803143 1332583531 1730977628 -1357265270 -1306992001
## [491] -346313783 -1348200354 -1861088500 1225626333 1375615975
## [496] 1324802064 -2071622354 -1752661541 2104039101 110609450
## [501] -798935864 2088153585 -248594589 1375438692 -1342710126
## [506] 339680103 339412273 -1709793674 -1153237100 967023525
## [511] 1441197231 -669851288 -2082485946 -1954455309 -1593581995
## [516] 678806066 793292800 -809806615 2002940763 -704399956
## [521] -1474171462 619207535 1144429817 -454344306 1376654428
## [526] -702241075 439022135 -360646464 1020250014 1394150507
## [531] 1226648557 1830891098 141685016 364494465 -589949901
## [536] 574661204 -696932766 -69744393 -1189074111 -139173434
## [541] 959818500 -1293869387 -222164577 -2044262920 -629527594
## [546] 936302179 1984759333 -618654270 -507901264 476381913
## [551] 1713189963 1381337788 1150826730 -143272673 -2072150615
## [556] -1880868034 1108170284 1786733693 -337355065 1231308720
## [561] -241100338 -1315900677 -638406307 973812362 493150376
## [566] 631473169 -2144865021 110336004 -1870784462 959393927
## [571] -2044797743 1762772566 2137167412 -1200329403 -2042372209
## [576] -1338684664 -42431194 818738643 300465973 940430994
## [581] -2119839136 -2032286775 -528121093 1230630092 -1326619302
## [586] 1459775567 -546757351 1137938222 293126460 551896749
## [591] -784432809 -2068451168 -1197928962 1449124043 -953696563
## [596] -21919238 -354282952 294997985 -287995757 1958161076
## [601] 499926978 48448471 687819809 -161313818 2065618660
## [606] -1544879595 1250609727 956842456 -828770314 1581785091
## [611] 381127493 -642424990 -781451056 650030073 -1559066837
## [616] 1326077084 1958283978 -766557249 61402889 -877627362
## [621] 371501516 809100189 245440039 924742736 -157008658
## [626] 2136795264
```

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

```
## Warning: package 'rpart' was built under R version 3.4.4
```

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 information 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 separate datasets: *train* and *test*. Both datasets contain details about passengers and their trip. The only difference 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 predictions, 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.

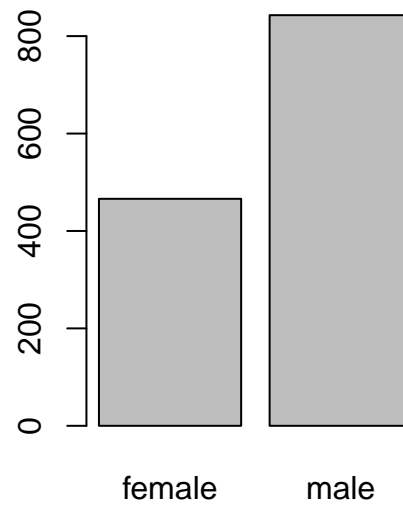
Name

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

example: *Allison, Master. Hudson Trevor*

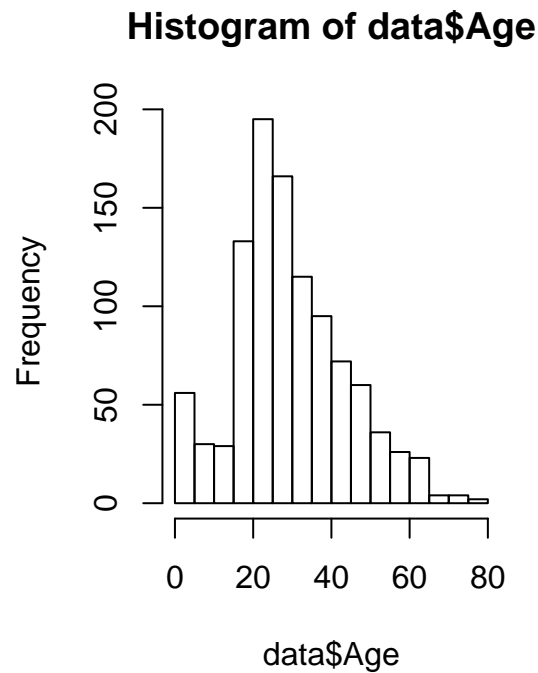
Fortunately, all rows in *Name* column follow the same string pattern (*surname, title first name*). Thanks to this fact, we will be able to retrieve more additional infrmation about passengers, like common surnames or titles.

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 distinguish 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 could result in better prediction 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 same 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 predictions.

Our assumptions:

- the number of relatives with who each passenger was traveling is calculated as follows: $SibSp + Parch + 1$ - result is family size
- if family size is less or equal 2 we assume that the value is not relevant and we mark such a family 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 that it's a significant 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 or 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 situation we decided to use a Decision tree to predict missing Age values. As significant factors we marked attributes: Pclass, Sex, FamilySize, Embarked, Title, SibSp, Parch.

Also Fare column had some missing values. In such a case we replaced 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 columns we decided that only few of them play significant role in predictions.

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

Creating a setup

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 predictions
- 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 submitted 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 satellite images. Each image had a resolution of 256 by 256 pixels, and covered an area of approximately 950 by 950 meters. The satellite images were captured above the Amazon in South-America. The goal of the competition was to successfully classify these satellite 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

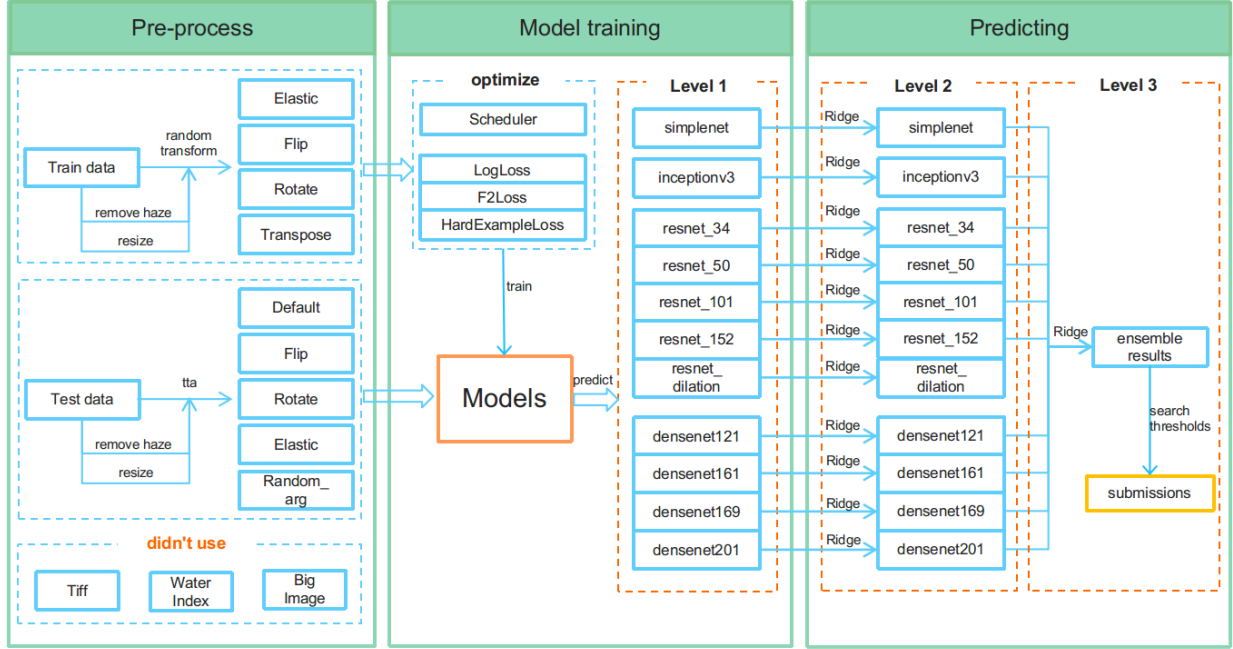
The participants’ model predictions were evaluated with their mean F2 score. This score determines the model prediction accuracy using the precision and recall measures. The formula of the score:

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

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 satellite 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 prediction.

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) and Mean Absolute Error(MAE)

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

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

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

Where, \hat{Y}_i is a vector of n forecasts 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 accuracy 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 forecasted 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 (X_1, X_2, \dots, X_8) along with two response variables (Y_1, Y_2):

- Relative Compactness(X_1) * Surface Area(X_2) * Wall Area(X_3) * Roof Area(X_4)
- Overall Height(X_5) - Orientation(X_6) * Glazing Area(X_7)
- Glazing Area Distribution(X_8)
- Heating Load(Y_1) * Cooling Load(Y_2)

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 `.sample()`. 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 attributes 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 interested 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 `.sample()` 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 vary 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 variables 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 Collection

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 probability model. It is more suited to categorical 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 `.sample()`. 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 its 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 splitted 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.

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 distinctive 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 classification 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)