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:

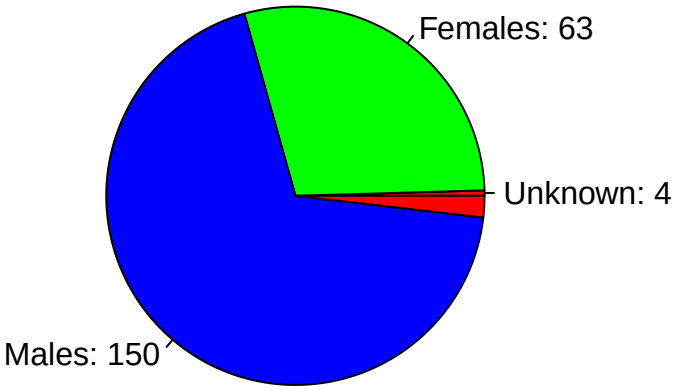


Figure 1. Pie chart showing the gender distribution in the data mining techniques course.

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

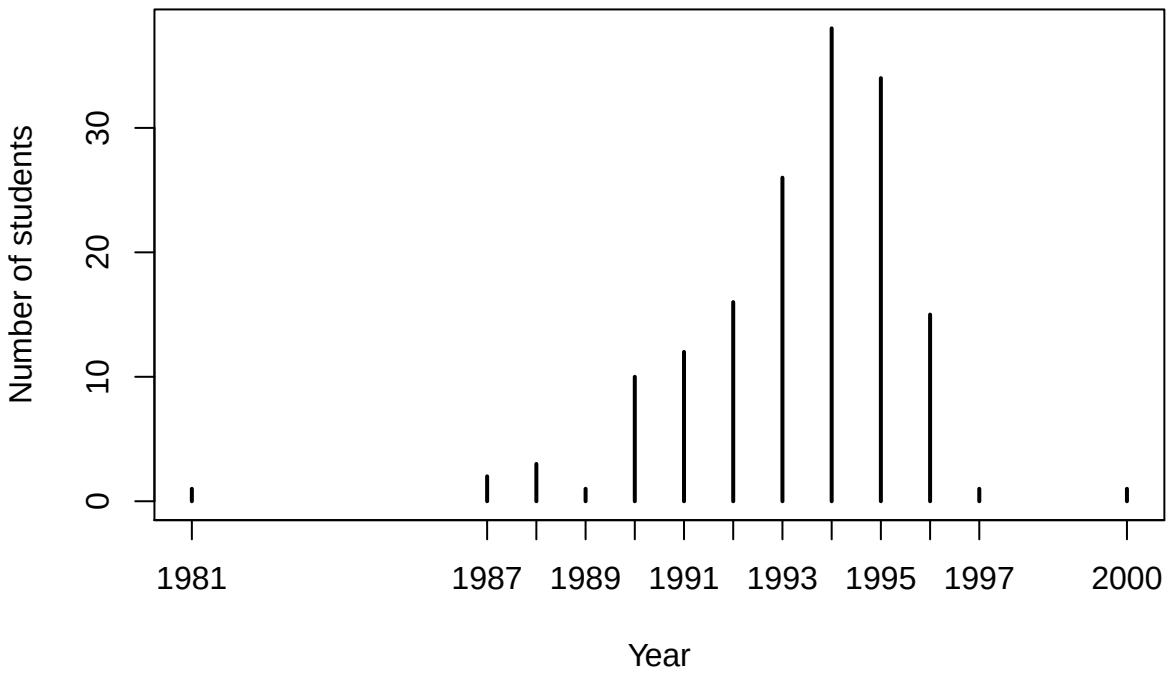


Figure 2. Birth year distribution in the data mining techniques course.

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.

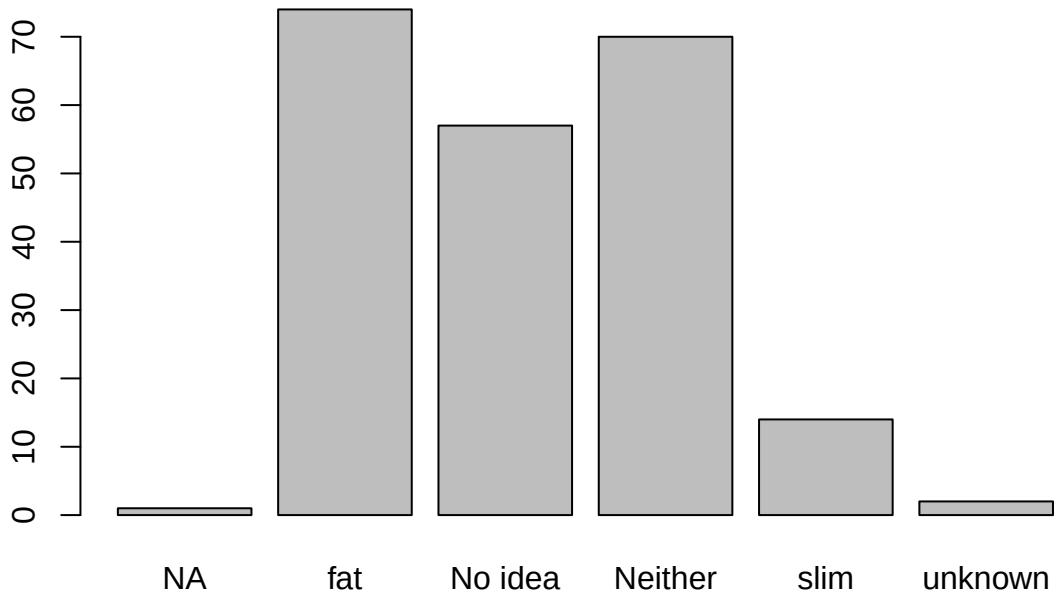


Figure 3. Chocolate makes you.... Answer frequencies.

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   967918199  1641692062 -1657869736
##   [6] -1026571333 -1699543719   315055848 -2056626410  -345902959
##  [11]  1545293123  -626157486   678767332  -290034393  -510052995
##  [16]  -574907276 -1004829094  -193740123   139724991  1604172614
##  [21] -1518138928  1168710643  -689224943   827642688    99351326
##  [26]   227034089 -1611544293 -2073959638  1236542540  1528507407
##  [31]  -161118395 -1562169092 -1281786030   834400461  1944912007
##  [36]  1298260846   330601800   541172043 -1379724119  1586095608
##  [41]  -152875738 -1283936703  1736189907 -1503560318  1019285940
```

```
##  [46]  -450839241   992802861  -839654076     5431914  2033257685
##  [51]  2124597871   -37191818 -2123546464 -2095687325  -510027263
##  [56]   137628016  -790480626 -1846932615   927793931  1223593018
##  [61]  1786817340   465961087  1651123157  2035895788  -953524414
##  [66]   766599325  1394071831  1103802494  1750534200  -570220773
##  [71]   -18224007  -305597240  1835275510  2025391409   -53725853
##  [76]   891606834 -1404108668   -50963257 -1100460131  1373534996
##  [81]  2069629370   557686789 -1312172577  1294837862  1093375984
##  [86]   -51191533   513425073 -1915487904  -390312194 -1829885303
##  [91] -2055518789  1712177034   457680492 -1304553489  -433803227
##  [96] -2132655908  1456088754   543377837  1632648551  -686790002
## [101] -2038402200 -1310348437   223974665 -1184142440 -1803752762
## [106]   569757729 -1936747341 -2086631902  -356304876   608615575
## [111]  2055214733  -193366108  -326669942 -1168353803   486918223
## [116]  1286659926  1160427008   339478339    -3940639 -2030027696
## [121] -1720548178   969973849 -1282424725 -1598196262   171302556
## [126]   633335967 -1224635659 -1549002868  -380029342  -260666819
## [131]   723432631   816769886 -2023240040  1395109371  -838340711
## [136]  2111522856 -1118916394  -416058031  1320035331  -121615470
## [141]   429828388  2143252455   864678973   760529588  2006798618
## [146] -1993465243  -188479745  -123372666   823254288   884921779
## [151]  1969260113  1117054464  2041586398   962244905  1957666011
## [156]  1774267370  -662668404  -767214385   958194821  1186311996
## [161]   355085842 -1783419635  1129805383   916597806  -516017784
## [166]   345172107  2143322729 -1669792200  -572090650  -489835903
## [171]  1022901779  -386446910   217277044   349865079   361975149
## [176]   215937540   745066794  1112671893   515075375  -639576010
## [181]  -671486880 -1557495133   755283393  -527072848  -840441650
## [186]  1831140921 -1523892149  1528427002  1319644412   206815551
## [191]   935575061    52722988   -68989438 -1553607459 -1522479145
## [196]   491170494   602713336  -402386213  2027997497  1118145672
## [201] -1840710602    19187825   318601891  1369169906   817951556
## [206]  -357344249 -1485112867   307119828  1627613690  -730806971
## [211] -1222635105  1022178086 -1464726608  1361049683  -306482319
## [216] -1593113952 -1619790274  2142502217  1064584187  1223108554
## [221]   618535212   153924911 -1799537691  2099741340  1602012914
## [226] -1612654739  1097113767  1161241294 -1768271832   413632146
## [231]  -722286816  -617515060  1143128288 -1306394942   -69553112
## [236] -1247405140    94583484   743311858  1322309168  1818179492
## [241]  1640722872 -1012694182  1591342224  -388093060  1011345172
## [246]  2128399682  -452694272  -408824900  -909258928 -1685583582
## [251]  -450469304  2068385804 -1913072548  -111955502  1953283584
## [256]   494838100 -1240117016  -127049478  1666107088  -729886420
## [261] -1907203852    36998546 -1877190304  1570985996  -218709280
## [266] -1810978206 -1128512856  -682694068   445214588   857150770
## [271]    19392112  2067166020   674785240   731187066    46446128
## [276]   859203516 -1474775468  1157124162  1246654624  -966347076
## [281]   741435856  -329104542  -953769240  -244072244   230121980
## [286]  1266545138  -211547520  1609473524  -635652472   852320058
## [291] -1271409840  -205788052 -2058747500  1621800018  -499121696
## [296]   490462412  1692484768  -492898942  1164757224   976320236
## [301] -2033887940  -736555918  -744888208  -536641500  1502989368
## [306] -1931301094 -1342870320  1970847548  -553261164 -1478863998
## [311] -1510520384   606895676  1576781200  1555250722  1152689416
```

```
## [316]  -304009588  1643683612 -2132623214 -1277249152   810051092
## [321]   602775336  -682026438  1955151312   788081004  -773585996
## [326]   614357010  1224472032 -1148532788   160424160   855808034
## [331]  1004298536 -1156342260 -1593465028 -1164917774   526392048
## [336] -1655288252 -1387281832  1283018554  1703142896  -189442372
## [341]   268220692   918176450  -650087328   489161596  -538306608
## [346] -1250598622  1172306600  -339502196  -715867972  -572002510
## [351]  1520151104  2100231348 -1624694200 -2049514054  1826521616
## [356] -1313934612  2123342164  1711718674  -644690528  -591748404
## [361]  -250929568 -1119717438 -1962842328  -521572436   461038396
## [366]    75213170  1936390960  1067511076  -337138888    14196442
## [371] -1847908464  1965313788  1692773268 -1750163390  1278334208
## [376]  1626965052  -780877488  1527877794  -567221944  1904809740
## [381] -1827683620  1552379218  -564149120  -931389356   232394728
## [386]    68954362 -1355273392  1176633132   669985140  2012277650
## [391] -1531402144 -1611739764 -1414505632   180500962  1462858664
## [396]  -162777908  2144141308  1199302066   127989232  1715571652
## [401] -1123998376   927750522 -1050082384   948799932  -488163244
## [406]   898873282   263335968 -1386399556  1401655888  1998631906
## [411]   819363944  -146389940 -1184818948 -1890808846 -2076260096
## [416]   683098484 -1743131640   696680378 -1874054704 -1015514132
## [421]   295182740  -926477870 -1101655712   725471052   612129056
## [426]   465485698  -825577752 -1419034388  -685558724   810236786
## [431]  -947372432  2107610148  1186876216 -1047221094 -1758260528
## [436]  1256292540  1588276884 -2117192062  -567588672  -661249220
## [441] -2087134064 -1927217630  1746128904  -874144500 -1381225316
## [446]  1304755730  -554008704  2100186644 -1052792280 -1638959814
## [451] -2067379760  1856844908   132589876  1872860818 -1310880928
## [456] -1001327796 -1201939111 -1865019701  2073585468  1067315050
## [461]   371482015  -618499031   610035390  1225557932 -1490907907
## [466]   136050759 -2049915088  1528209486   602647675   503807965
## [471]  2068049418   974500648  -173736559  -542653565   268413060
## [476]   -15841614   284645895 -1582716335  -568057386 -2010919756
## [481]  1237694661   691980815  -628317560  -892774490   247144275
## [486]  1445290165   679099922 -1510286368 -1718751671  -750916485
## [491] -1444901556  -841074470  1874413775  -529853287  1346885806
## [496]  1199998396  -713744851   786856407 -1863217120  1664775550
## [501]  -857937077   171618637 -1955295366 -1358700104 -1313307807
## [506]  1296438035  1830087476  2078566722   306195799  -290043743
## [511]   658579814  1962726244   -30213483 -1495909441  -610889896
## [516]  -134625930  -262527101 -1330577979  1182892770 -1658999984
## [521] -1526614407 -1872676437 -1997304036  1106008138   158616639
## [526]  1202900105  -178607970  -298932916  2105836829  -427540313
## [531]  1243688912  1659791214    16485915 -1163004035 -1611039382
## [536] -1385245688  2076302513  -278725853 -1141110748   428673746
## [541]  -279453913   905216369   576309174  1482079060  1970849253
## [546]  -377274385 -1972920536  1291298694  1272943539   296416021
## [551] -1442837134 -2111344192 -1890557399  1075095195  1088188268
## [556]  1821931386 -1808068433  2118429625  1781513678  -240490212
## [561]  1002953869   256594167   724392960  -331784610 -1734557397
## [566]  1417257517   793977498  1909546584  -377495103 -1453855117
## [571] -1947407468 -2134973918   822471991   888572929  1562947846
## [576] -1757751996  1058338933  1832638943 -2133007048  -544245994
## [581]  1345877795  1251149029   172146946  1884795760  -786945255
```

```
## [586] -1127560437  1674385404 -1107332054   388110431 -1768763415
## [591]   779170046   834494700 -1170930627   324827143  -181435408
## [596] -2080813170 -1899999813 -1590416483  -214257334   850057576
## [601]   400019281    74173507 -1522494140  -261114382  -560742457
## [606] -1696071279 -1385057002 -1745400460  -328119291 -1046922161
## [611]  1420972104   896517606  -201909485 -1305969547  -292624814
## [616]  1466867872  -612927095  -274728517   966383116 -1409298534
## [621] -1936820465   510389593   433063918  1012133500  -709372179
## [626] -2106530796
```

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: Compete in a Kaggle Competition to Predict Titanic Survival

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

For further processing we decided to combine both datasets into big one. Such an approach allows us to perform more adequate data analyse as we have a full insight of passengers.

Data exploration

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 - indicates if given passenger 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 (male or female)
- 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)

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 additional information about passengers, like common surnames or titles.

Sex

According to data, there were 466 females and 843 males onboard. That gives us the first easy grouping of passengers. According to the rule "women and children first", Sex could be siginificant attribute in predictions.

Age

Regarding Age attribute, we can see that this variable varies from 0.17 up to 80, with mean around 29.7. Age attribute can also be considered as significant factor. Do young people are more likely to survive?

Feature Engineering

During feature engineering 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').

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'.

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 (70% - training, 30% - testing).

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:

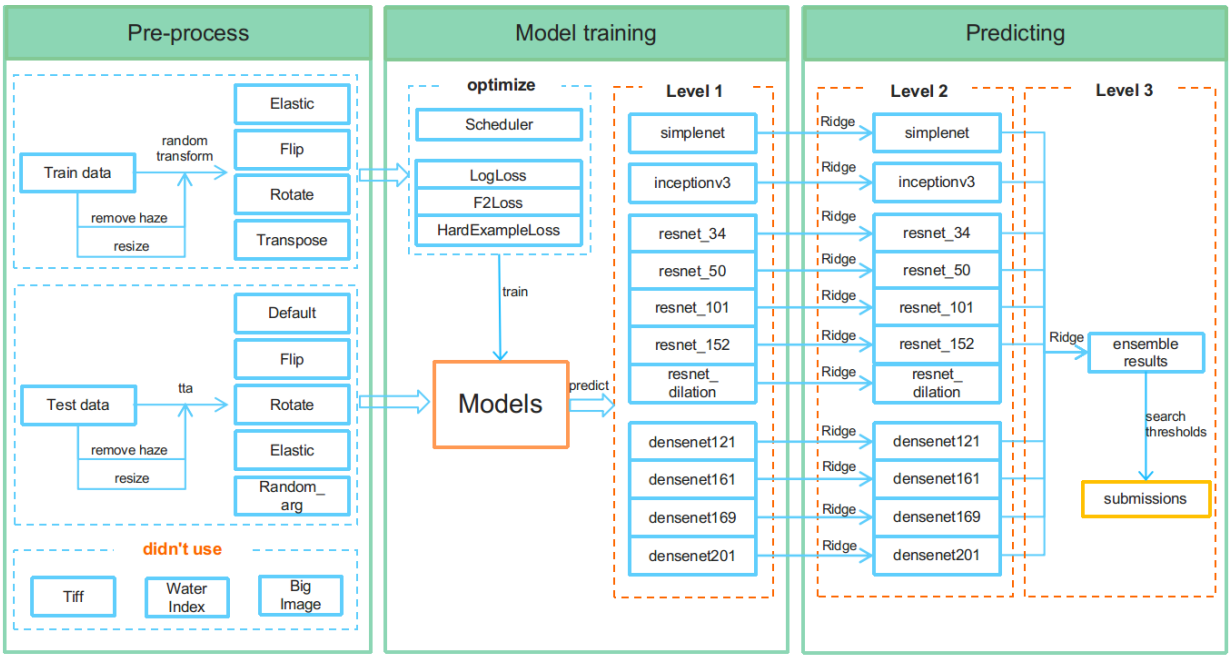
$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \quad \text{where} \quad p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}, \quad \beta = 2.$$

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) 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 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.

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)