SUPERVISED LEARNING METHOD-TITANIC DATASET

ABSTRACT

This project is to use supervised learning methods to do complete the analysis of what sorts of people were likely to survive. In particular, this analysis is done to apply the tools of machine learning to predict which passengers survived the tragedy. Different type of machine learning techniques like decision tree, logistic regression, naïve bayes and Knn was performed and the technique with best accuracy is chosen.

GOALS AND METHODS OF THE REPORT

The main aim of this report is to analyse a dataset containing mixed type variables (continuous, binary and categorical) with a binary response by means of several different supervised learning methods, in order to find the most suited ones to make predictions on new data. The analysis will focus more on predictive capability rather than model interpretability. The models considered for this analysis are:

- Decision tree
- Logistic Regression
- Naïve Bayes
- KNN

The study will include the theoretical background of the supervised learning methods used for the analysis. The performance of each approach is evaluated and then the best model for the prediction is considered.

DATA VARIABLES AND DESCRIPTION

The Titanic passenger data consists of a training set and a test set, both of which are .csv files. The training set includes the response variable Survived and 11 other descriptive variables pertaining to 891 passengers. The test set does not include the response variable but does contain the 11 other variables for 418 passengers. A few additional notes were made on the competition page regarding specific details for some of the variables encountered in the Titanic data. It is first noted that the Age variable is measured in years but can also appear as a fraction if the passenger is less than one year old.

Furthermore, one will be able to tell if Age was estimated, that is if the age value is followed by.5,e.g. 28.5.There was also further explanation provided for the family relation variables, i.e. SibSp and Parch.It may be of importance to note how these variables were defined and any possible exclusions based on these definitions. SibSp is an abbreviation for siblings and spouse. Siblings accounted for by the SibSp variable are brothers, sisters, stepbrothers, or stepsisters aboard the Titanic. The typical categorizations of a spouse, i.e., husband or wife, aboard the titanic are captured by the SibSp variable as well. Thus, any fiancés, mistresses, or the like are not included in the SibSp variable. The Parch variable identifies both parents and children for each passenger aboard the Titanic. Parents are then considered to be either a mother or father.

Children can be a son, daughter, stepson, or stepdaughter. Based on these definitions, one can conclude that family relatives such as, cousins, nephews/nieces, uncles/aunts, and in-laws are not captured by the family relation variables. Furthermore, if a child made the voyage with a nanny only, or neighbours, or friends of the family, then the Parch variable will be equal to 0.

Variable Name	Variable Description	Possible Values	Categorical/Numerical
PassengerId	Observation Number	1, 2, …, 1309	Numerical
Survived	Survival	1 = Yes, 0 = No	Categorical
Pclass	Passenger Class	1 = 1st, 2 = 2nd, 3 = 3rd	Categorical
Name	Passenger Name	Braund, Mr. Owen Harris, Heikkinen, Miss. Laina, etc.	Categorical
Sex	Sex of Passenger	Female, Male	Categorical
Age	Age of Passenger	0.17 – 80	Numerical
SibSp	No. of Siblings/Spouses Aboard	0 – 8	Numerical
Parch	No. of Parents/Children Aboard	0 – 9	Numerical
Ticket	Ticket Number	680 – 3101298, A. 2. 39186, WE/P 5735, etc.	Categorical
Fare	Passenger Fare	0 – 512.3292	Numerical
Cabin	Passenger Cabin	A10, B101, C103, D, E12, F2, G6, etc.	Categorical
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton	Categorical

The target variable is ‘survived’ which indicates if people survived or not in the tragedy.

FINDING AND KEYS

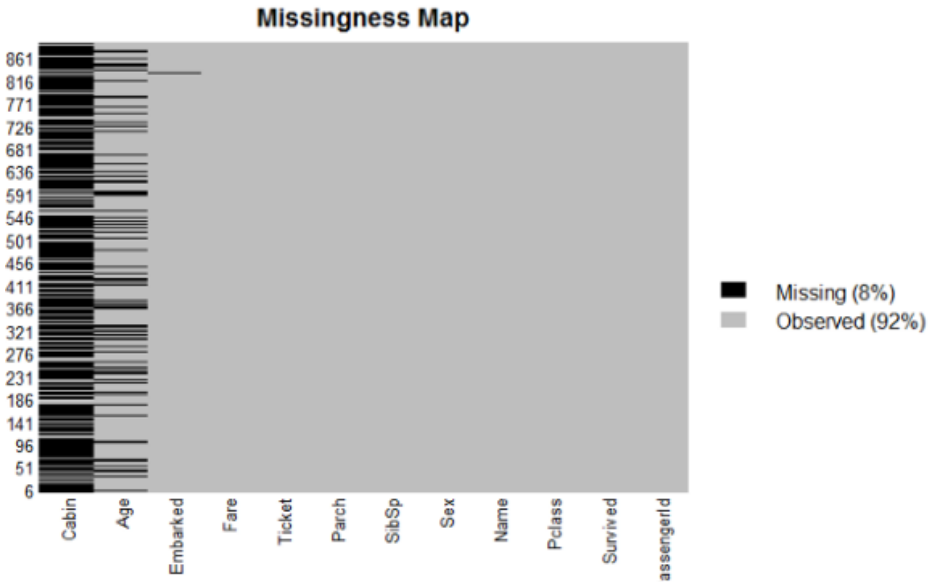
The chances of survival for passengers in 1st class was more than the others.

- Most of the passengers were in age group of 20 to 40.
- A smaller number of people survived and in those more number of females survived than males.
- The accuracy obtained from KNN was found to be 85.3%
- The accuracy obtained from Naïve Bayes Model was found to be 81.81%
- The accuracy obtained from Logistic regression was found to be 80%
- The accuracy obtained from Decision tree is 81.11% and after fine tuning was found to be 87.41%
- sex is a good predictor for the survival of passengers.

DATA CLEANING AND PREPROCESSING

Before starting the actual analysis, it is necessary to inspect and prepare the dataset, according to one's needs. First the target variable 'Survived' since it was a categorical variable in the data it was changed to a dichotomized as 0,1.

The next step was to remove the missing variables which are of no use and can cause errors while doing the analysis. By visualising the dataset in the form of missingness map. in order to avoid removing too many instances, it is better to directly remove the entire column. The variables which present the highest number of NA value is 'Cabin' ,'PassengerID' therefore they are dropped. Some variables which did not decide the response variable 'survival' they were also dropped. 'Survived' and 'Pclass' column are integers but they are actually categorical, so they were converted using factor function.



After removing the missing values, it was again checked by plotting the missingness Map

DATA VISUALISATION

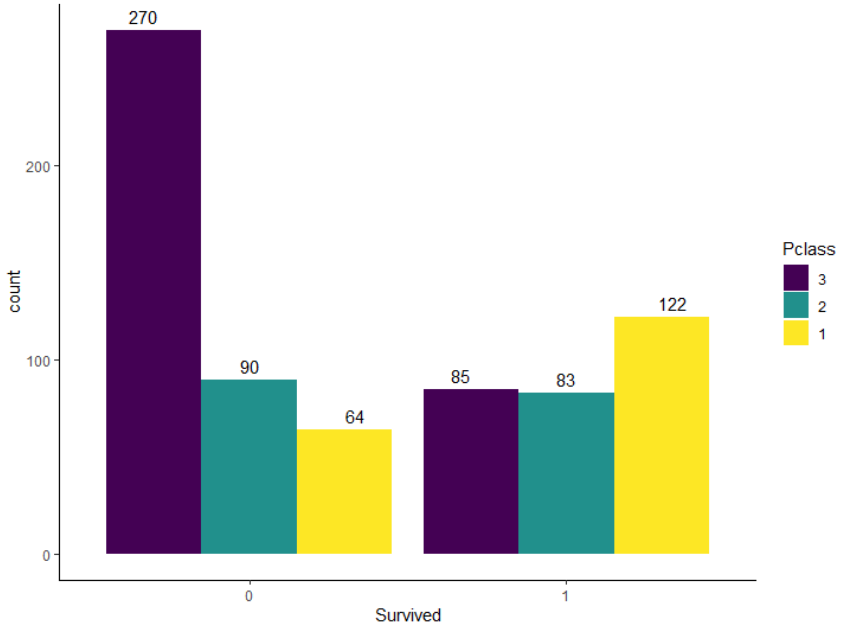
Data visualisation was performed to find the type of passengers who survived or died in the titanic tragedy.

First the correlation matrix was considered to find the correlation between all the variables

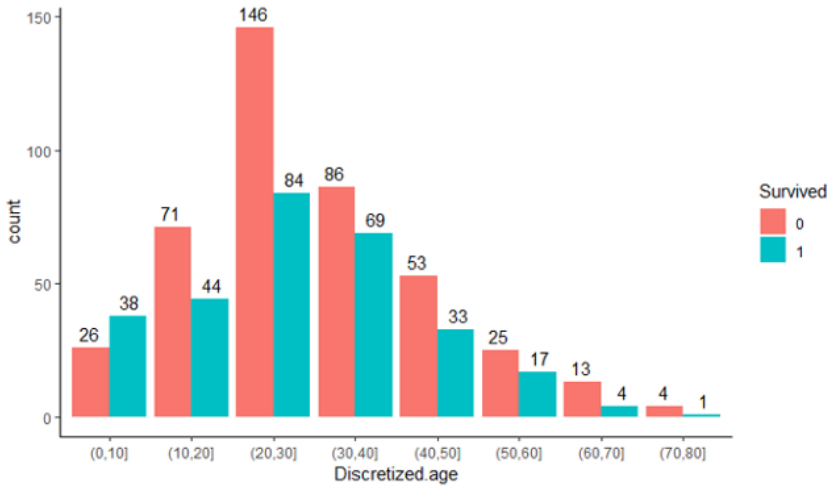
Then count of survived or died people were found.

The survival count was found by sex to find how many male and how many female survived and it was found that very less people survived.

The survival count was done with class to find out which class of passengers were having more chance of survival compared to others and the first class passengers had more chance of living compared about others.



The survival rate by age plot was also done to check which age group of people had more chance of living compared to other age groups. And found that most passengers were 20-40 age, Children less than 5 years had higher survival chances. Passengers aged 20-40 were more likely to die. Passengers aged about 65 - 75 had an almost 0 survival chance. One passenger aged 80 years survived.



DATA ANALYSIS

The methods of supervised learning is used in the analysis. Different types of methods were performed to find the method with best accuracy

The data is divided into train and test sets. A fraction argument is passed which determines the fraction of records that must be selected. And then accuracy is calculated.

LOGISTIC REGRESSION

Due to the simplicity first model considered is Logistic regression.

Logistic Regression is used to obtain odds ratio in the presence of more than one explanatory variable. The procedure is quite similar to multiple linear regression, with the exception that the response variable is binomial. The result is the impact of each variable on the odds ratio of the observed event of interest.

The logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression since the outcome in logistic regression is a probability between 0 and 1. The weights do not influence the probability linearly any longer. The weighted sum is transformed by the logistic function to a probability. Therefore, we need to reformulate the equation for the interpretation so that only the linear term is on the right side of the formula.

$$log\left(\frac{P(y=1)}{1-P(y=1)}\right) = log\left(\frac{P(y=1)}{P(y=0)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

The term in the log() function "odds" (probability of event divided by probability of no event) and wrapped in the logarithm is called log odds.

This formula shows that the logistic regression model is a linear model for the log odds.

The logistic regression model is not only a classification model, but also gives us probabilities.

In order to estimate the coefficients, the maximum likelihood method is used, which finds values of b such that they maximize the likelihood (probability) that the observed data were actually produced by the fitted model. It maximizes the probability that our model could have generated the observed data. Another advantage of logistic regression is that when fitting the model, the interpretation of the found coefficients is analogous to the linear regression, since what it is shown in the output is the log-odds. For continuous variable, an increase of 1 in a predictor means an increase of b in the log-odds.

```
Call:
glm(formula = Survived ~ ., family = "binomial", data = r_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5920  -0.6739  -0.4134   0.6367   2.3344

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.419361   0.196762   7.214 5.45e-13 ***
Pclass.L     1.555536   0.228015   6.822 8.97e-12 ***
Pclass.Q    -0.063754   0.217026  -0.294 0.768940
Age         -0.532113   0.138546  -3.841 0.000123 ***
Sexmale     -2.569211   0.251982 -10.196  < 2e-16 ***
SibSp       -0.316053   0.135620  -2.330 0.019784 *
Parch       -0.009988   0.127804  -0.078 0.937709
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 677.21  on 498  degrees of freedom
Residual deviance: 461.88  on 492  degrees of freedom
AIC: 475.88
```

From the output it can be seen that only few variables are insignificant.

Intercept=1.419361

Parch =0.937is insignificant so null hypothesis is not rejected (P>0.5)

Pclass.Q= 0.768 is insignificant so null hypothesis is not rejected (P>0.5)

Given p<0.5, we can reject the null hypothesis that one unit increase in age does not affect chances of survival.

For every unit increase in Age the log-odds of survival decrease by -0.5 i.e., the chances of survival decrease as passenger age increases.

The difference in the log-odds of survival between men and women is -2.569 i.e., the chance of survival is lower for men than for women.

For every unit increase in Age the log-odds of survival decrease by -0.5 i.e., the chances of survival decrease as passenger age increases.

For every unit increase in SibSp the log odds of survival decrease by -0.31 i.e., the chance of survival decreases as number of Siblings/Spouses Aboard increases.

If the aim of the analysis would have been to interpret the results, the number of regression predictors should be reduced to get rid of non-significant features. They cannot be simply eliminated by looking at their significance, since some variables may look significant or nonsignificant due to collinearity, even though actually it is the opposite. In this case the analysis is more concerned with obtaining a good prediction, so the non-significant features can be left as they are

The performance of the logistic regression will be evaluated as follows. When assessing the goodness of a model fitted on a certain dataset, it is more informative to look at the test error, which is the error in predicting new data, rather than the training error, which is, instead, the mistake made on the dataset used to train the model. What really matters is to have a model able to generalize, so that it can be used to make predictions and explain more general settings. The test error plays a pivotal role in choosing the correct flexibility of the model, namely the number or value of parameters for parametric models. A good model finds the optimal trade-off between bias and variance error, so that it is able to fit data properly without being affected too much by noise. If a model is overly flexible, it will fit the training data well but the new data will be fitted poorly, leading to overfitting. On the other hand, if it is not flexible enough it might perform poorly on both datasets, leading in this case to underfitting. Another important aspect, other than model flexibility, is to always choose more parsimonious models when adding more parameters doesn't increase model performance, so that the model is easier to interpret. The training error cannot directly be used as an approximation for test error, since the two often differ. The training error is always decreasing as the flexibility of the model increases, while the test error has a typical u-shape, first decreasing due to the effect of variance reduction, then increasing due to the increase in bias, which after a certain point dominates the decrease in variance.

To evaluate the performance of a logistic regression model, Deviance is a number that measures the goodness of fit of a logistic regression model. Think of it as the distance from the perfect fit — a measure of how much your logistic regression model deviates from an ideal model that perfectly fits the data. Deviance ranges from 0 to infinity.

Null Deviance indicates the response predicted by a model with nothing but an intercept.

Model deviance indicates the response predicted by a model on adding independent variables. If the model deviance is significantly smaller than the null deviance, one can conclude that the parameter or set of parameters significantly improved model fit.

Another way to find the accuracy of model is by using Confusion Matrix.

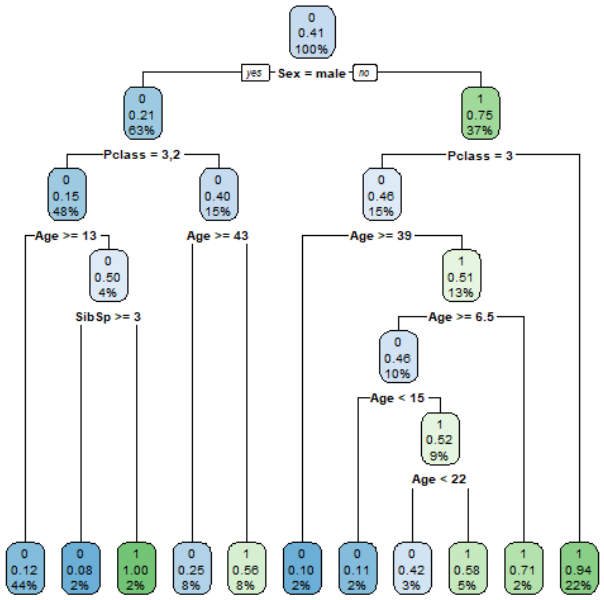
The accuracy of the model is given by:

$$\frac{\text{True Positive + True Negatives}}{\text{True Positive + True Negatives + False Positives + False Negatives}}$$

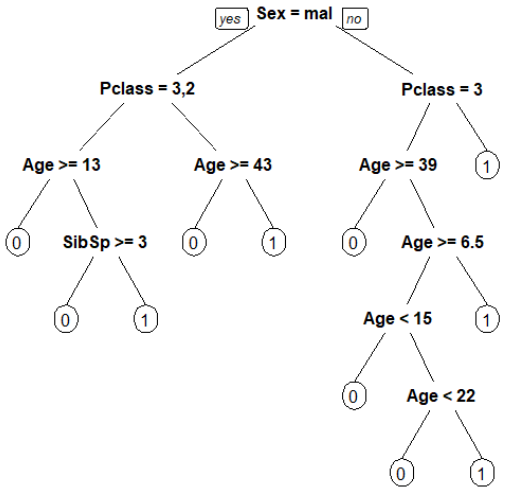
DECISION TREE

The Decision Tree is a very robust supervised machine learning algorithm implemented for classification and regression problems. Thus, it can be used for continuous and categorical variables to make a tree. Consequently, this is one technique to show an algorithm that only includes claims for conditional monitoring.

Decisions trees are useful for making ‘decisions’ about what is happening in data



The Titanic survivor dataset was used to grow a Decision Tree. In it, each node-branch can be interpreted as an if-then rule, where if (node) then (branch), given observed probabilities during the training phase. In our Decision Tree, Sex is the most information gaining (opposite of entropy) feature, followed by Ticket Class , Age, SibSp and then Parch.CP is the complexity parameter. It prevents overfitting and controls the size of the tree. To get added into a node, a variable has to be having CP less than 0.008 or else tree building will not continue..



A total of 11 nodes got created. There are 6 leaf nodes representing class 0 and 5 leaf nodes representing class 1.

Each node shows predicted class 0 for not survived and 1 for survived.

The predicted probability of survival

The percentage of observations in each node

ACCURACY

```
> table
   predicted
      0  1
  0 71 16
  1 11 45
```

(TN) (FP)

(FN) (TP)

After training the model, w to make predictions on the test set using predict() function. the fitted model was passed with the test data and type = 'class' for classification. It returns a vector of predictions. The table() function produces a table of the actual labels vs predicted labels, also called confusion matrix

The accuracy is calculated using (TP + TN)/ (TP + TN + FP + FN) The accuracy was found to be 81.18%.

FINE TUNING

The fine tuning of the decision tree was done with the control parameter by selecting the minsplit(min number of samples for decision), minbucket(min number of samples at leaf node), maxdepth(max depth of the tree).

minsplit is "the minimum number of observations that must exist in a node in order for a split to be attempted" and minbucket is "the minimum number of observations in any terminal node"

Now let's grow our decision tree, restricting it to one split by setting the maxdepth argument to 6

In our decision tree we have used

(minsplit = 8, minbucket = 2, maxdepth = 6,cp = 0)

The two variables minbucket and minsplit are closely related. In rpart if either is not specified then by default the other is calculated as $minsplit = 3 * minbucket$

Whilst the default is to set minbucket to be one third of minsplit there is no requirement for minbucket to be less than minsplit

The variable cp governs the minimum complexity benefit that must be gained at each step in order to make a split worthwhile. The default is 0.01.

The complexity parameter (cp) is used to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue. We could also say that tree construction does not continue unless it would decrease the overall lack of fit by a factor of cp.

Setting this to zero will build a tree to its maximum depth (and perhaps will build a very, very, large tree). This is useful if we want to look at the values for CP for various tree sizes. This information will be in the text view window. we can look for the number of splits where the sum of the xerror (cross validation error, relative to the root node error) and xstd is minimum.

THEORITICAL CONCEPT

Decision tree are commonly referred to as tree-based models or decision tree methods for regression and classification settings Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. Rectangular regions. The term "decision tree" was coined to describe these methods considering that the collection of if-then statements used to divide up the predictor space can be represented using a tree graphic. Tree-based methods are widely used in statistical modelling due to having several advantages. In general, the application of tree-based methods is simple and the results are easy to interpret. As an added benefit, these methods "can effectively handle many types of predictors (sparse, skewed, continuous, categorical, etc.) without the need to pre-process them"

and without knowing the relationship that each predictor has with the response. Moreover, tree-based methods can accommodate missing data and automatically select the influential predictors for the model.

Tree models where the target variable can take a finite set of values are called classification trees. The goal is to create a model that predicts the value of a target variable based on several input variables. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. The arcs coming from a node labelled with a feature are labelled with each of the possible values of the feature. Each leaf of the tree is labelled with a class or a probability distribution over the classes. A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data. A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

NAÏVE BAYES

Naive Bayes algorithm, in particular is a logic based technique which is simple yet so powerful that it is often known to outperform complex algorithms for very large dataset

The reason that Naive Bayes algorithm is called Naive is not because it is simple or stupid. It is because the algorithm makes a very strong assumption about the data having features independent

of each other while in reality, they may be dependent in some way. In other words, it assumes that the presence of one feature in a class is completely unrelated to the presence of all other features. If this assumption of independence holds, Naive Bayes performs extremely well and often better than other models. Naive Bayes can also be used with continuous features but is more suited to categorical variables. If all the input features are categorical, Naive Bayes is recommended. However, in case of numeric features, it makes another strong assumption which is that the numerical variable is normally distributed.

R supports a package called ‘e1071’ which provides the naive bayes training function

The model creates the conditional probability for each feature separately. We also have the a-priori probabilities which indicates the distribution of our data.

The accuracy found is 81.81%

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.5901926 0.4098074

Conditional probabilities:
   Pclass
Y           3         2         1
  0 0.6290801 0.2077151 0.1632047
  1 0.2863248 0.2948718 0.4188034

   Age
Y       [,1]     [,2]
  0 31.09347 14.50516
  1 28.63996 14.61070

   Sex
Y       female      male
  0 0.1543027 0.8456973
  1 0.6752137 0.3247863

   SibSp
Y        [,1]      [,2]
  0 0.5578635 1.0678216
  1 0.5000000 0.7541018

   Parch
Y        [,1]      [,2]
  0 0.3946588 0.9071646
  1 0.5000000 0.8036932
```

KNN

The kNN model is available in the ‘class’ library. But, knn() requires numeric variables. It throws error if factors are used in the data frame.

knn() accepts only matrices or data frames as train and test arguments and not vectors.

The knn () function needs to be used to train a model for which we need to install a package 'class'. The knn() function identifies the k-nearest neighbors using Euclidean distance where k is a user-specified number.

First dummy() was used to create a one-hot encoding for Pclass and Sex attributes. The original factor attributes are dropped. The train, test features and labels are separated, and the Survived attribute is dropped from the train, test set.

The value for k is generally chosen as the square root of the number of observations.

The accuracy obtained was 86%

The performance of the model is evaluated by using the cross table.

```
> CrossTable(x=test_labels,y=knn_predict,chisq=FALSE)

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  143

              | knn_predict
 test_labels |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
            0 |        84 |         3 |        87 |
              |     8.817 |    20.505 |           |
              |     0.966 |     0.034 |     0.608 |
              |     0.840 |     0.070 |           |
              |     0.587 |     0.021 |           |
-------------|-----------|-----------|-----------|
            1 |        16 |        40 |        56 |
              |    13.698 |    31.856 |           |
              |     0.286 |     0.714 |     0.392 |
              |     0.160 |     0.930 |           |
              |     0.112 |     0.280 |           |
-------------|-----------|-----------|-----------|
Column Total |       100 |        43 |       143 |
              |     0.699 |     0.301 |           |
-------------|-----------|-----------|-----------|
```

(TN) (FP)

(FN) (TP)

The test data consisted of 143 observations. Out of which 84 people have been accurately predicted (TN->True Negative) as died in nature which constitutes 58.7%. Also, 40 out of 143 observations were accurately predicted (TP-> True Positives) as survived(1) which constitutes 28%. Thus a total of 16 predictions where TP i.e., True Positive in nature.

There were 16 cases of False Negatives (FN) meaning 16 people who were alive which actually but got predicted as died. The FN's if any poses a potential threat for the same reason and the main focus to increase the accuracy of the model is to reduce FN's.

There were 3 cases of False Positives (FP) meaning 3 people actually died but got predicted as alive.

Accuracy can be calculated by 86.7%((TN+TP)/135) which shows that there are chances to improve the model

This can be taken into account by repeating the steps 3 and 4 and by changing the k-value. Generally, it is the square root of the observations and in this case we took k=10 which is near to the perfect square root of 143 because considering k=11 or k=12 didn't give a better accuracy.

CONCLUSION

The most relevant findings emerged from this study are summarized as follows:

- All the supervised learning models applied to the Titanic Dataset delivered a pretty good predictive performance.

- 2. The best model in terms of performance, model complexity and computational time is the Decision Tree model with 87% accuracy

- KNN performed well on the dataset but not as much as decision tree

- 5. The most significative variables in explaining the target variable (whether a people survived or not) seems to be ‘Pclass’ which is the ticket class, ‘Age’ which is the age of the passenger and ‘Sex’ which is the gender of all the passengers who travelled in titanic

APPENDIX

```r
data.frame = read.csv("https://raw.githubusercontent.com/AnjaliRajagopal/titanic/main/train.csv", header = T, na.strings = c(''))
library(psych)
library(ggplot2)
View(data.frame)
#finding the missing values of data
library(Amelia)
missmap(data.frame, col=c("black", "grey"))
library(dplyr)
data.frame = select(data.frame, Survived, Pclass, Age, Sex, SibSp, Parch)
data.frame = na.omit(data.frame)
str(data.frame)
data.frame$Survived = factor(data.frame$Survived)
data.frame$Pclass = factor(data.frame$Pclass, order=TRUE, levels = c(3, 2, 1))

# Histogram to find the survived people by different age groups
data.frame %>%
  ggplot(aes(x = Age, fill = Survived)) +
  geom_histogram() +
  theme_classic() +
  theme(
    plot.title = element_text(family = "Times New Roman", hjust = 0.5),
    axis.text = element_text(family = "Times New Roman",face = "bold"),
    axis.title = element_text(family = "Times New Roman", face = "bold"),
    legend.title = element_blank(),
    legend.text = element_text(family = "Times New Roman")

  ) +
  labs(title = "Survival rates by Age")

#plot to find the distribution of class among dead and survived people by gender

data.frame %>%
  ggplot() +
  geom_bar(aes(x = Survived, fill = Sex), position = "fill") +
  labs(title = "Distribution of classes\namong dead and safe passengers", y = "Proportion of passengers", x = "") +
  scale_x_discrete(labels=c("Died","Survived")) +
  scale_fill_brewer(palette="Blues")

#plot to find the survived people count

library(ggplot2)
ggplot(data.frame, aes(x = Survived)) +
  geom_bar(width=0.5, fill = "coral") +
  geom_text(stat='count', aes(label=stat(count)), vjust=-0.5) +
  theme_classic()
#plot to find surved people by gender
ggplot(data.frame, aes(x = Survived, fill=Sex)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat='count',
            aes(label=stat(count)),
            position = position_dodge(width=1), vjust=-0.5)+
  theme_classic()
#plot to find the survived people by each class of ticket
ggplot(data.frame, aes(x = Survived, fill=Pclass)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat='count',
            aes(label=stat(count)),
```

```
            position = position_dodge(width=1),
            vjust=-0.5)+
  theme_classic()

#Plot to find the density of age

ggplot(data.frame, aes(x = Age)) +
  geom_density(fill='coral')

#Discretize age to plot survival
data.frame$Discretized.age = cut(data.frame$Age,
c(0,10,20,30,40,50,60,70,80,100))
# Plot discretized age
ggplot(data.frame, aes(x = Discretized.age, fill=Survived)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat='count', aes(label=stat(count)), position = posi-
tion_dodge(width=1), vjust=-0.5)+
  theme_classic()
data.frame$Discretized.age = NULL

train_test_split = function(data, fraction = 0.8, train = TRUE) {
  total_rows = nrow(data)
  train_rows = fraction * total_rows
  sample = 1:train_rows
  if (train == TRUE) {
    return (data[sample, ])
  } else {
    return (data[-sample, ])
  }
}
train <- train_test_split(data.frame, 0.8, train = TRUE)
test <- train_test_split(data.frame, 0.8, train = FALSE)
#predicting decision tree
library(rpart)
library(rpart.plot)table
fit <- rpart(Survived~., data = train, method = 'class')
rpart.plot(fit, extra = 106)
predicted = predict(fit, test, type = 'class')
table = table(test$Survived, predicted)
dt_accuracy = sum(diag(table)) / sum(table)
paste("The accuracy is : ", dt_accuracy)

#tuning the decision tree and predicting decision tree
control = rpart.control(minsplit = 8,
                        minbucket = 2,
                        maxdepth = 6,
                        cp = 0)
tuned_fit = rpart(Survived~., data = train, method = 'class', control = control)
dt_predict = predict(tuned_fit, test, type = 'class')
table_mat = table(test$Survived, dt_predict)
dt_accuracy_2 = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", dt_accuracy_2)

#predicting logit regression

data_rescale = mutate_if(data.frame,
                         is.numeric,
                         list(~as.numeric(scale(.))))
r_train = train_test_split(data_rescale, 0.7, train = TRUE)
r_test = train_test_split(data_rescale, 0.7, train = FALSE)
logit = glm(Survived~., data = r_train, family = 'binomial')
summary(logit)
lr_predict = predict(logit, r_test, type = 'response')
```

```
# confusion matrix
table_mat = table(r_test$Survived, lr_predict > 0.68)
lr_accuracy = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", lr_accuracy)

#predicting naive bayes

library(e1071)
library(naivebayes)
nb_model = naiveBayes(Survived ~., data=train)
nb_predict = predict(nb_model,test)
table_mat = table(nb_predict, test$Survived)
nb_accuracy = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", nb_accuracy)

#predicting KNN

library(class)
library(dummies)
# one hot encoding using dummy
ohdata = cbind(data.frame, dummy(data.frame$Pclass))
ohdata = cbind(ohdata, dummy(ohdata$Sex))
# drop original factor variables
ohdata$Pclass = NULL
ohdata$Sex = NULL
ohtrain = train_test_split(ohdata, 0.8, train = TRUE)
ohtest = train_test_split(ohdata, 0.8, train = FALSE)
library(dplyr)
train_labels = select(ohtrain, Survived)[,1]
test_labels = select(ohtest, Survived)[,1]
# drop labels for prediction
ohtrain$Survived=NULL
ohtest$Survived=NULL
knn_predict = knn(train = ohtrain,
                  test = ohtest,
                  cl = train_labels,
                  k=10)
table_mat = table(knn_predict, test_labels)
accuracy_knn = sum(diag(table_mat)) / sum(table_mat)
#Evaluate the model performance
library(gmodels)
CrossTable(x=test_labels,y=knn_predict,chisq=FALSE)
```

DATA SHEET:

https://raw.githubusercontent.com/AnjaliRajagopal/titanic/main/train.csv