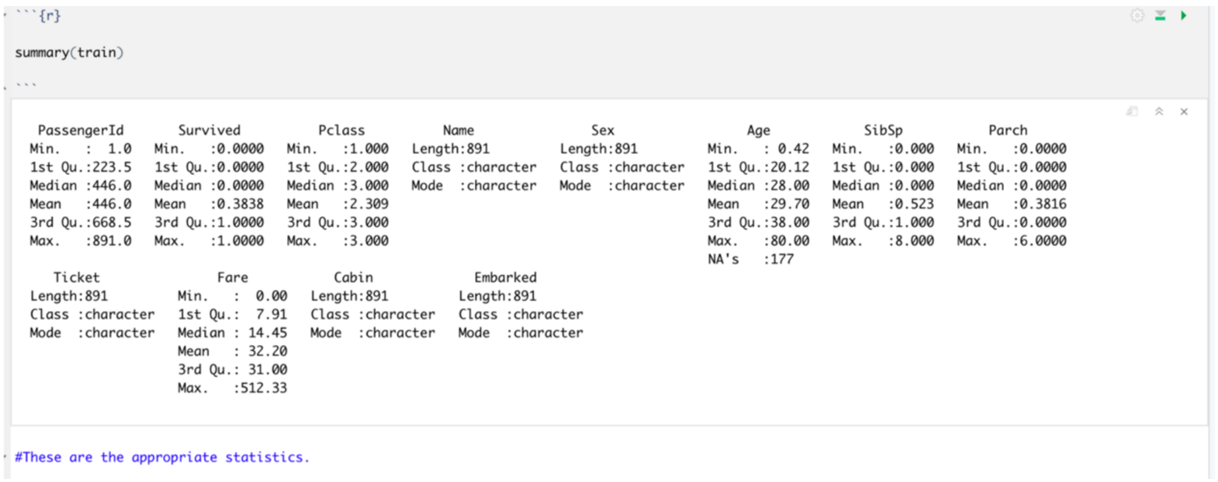
Introduction

For this Final project, I have chosen the Titanic problem - to predict the survival of Titanic Passengers. Moreover, I would need to predict what sorts of people were more likely to survive using the passenger data. The main data set is the training set. This gives us information regarding the outcome for each passenger also known as ground truth. There are various headings such as the passenegerID, whether they survived or not, the passenger class, the name, sex and age of the passenger, etc. According to this kind of information, we must come to a conclusion for our model regarding which kinds of passengers survived.

I would want to first categorize the data and see if there's any correlation of different classes. Next, I would want to convert any text categorical values to numerical values. Also, when preparing data for analysis, etc., we must always check to see if there are any missing values because model algorithms tend to work best when there are no missing values. I would also analyze the dataset to see if there are any errors. I would do this by checking the outliers or even the skewness. To measure the effectiveness of a good prediction algorithm or clustering algorithm, I would use cluster validity indices.

Explanatory Data Analysis

By using the summary function in R, we see the following statistics for the various attributes of the train data set. Here is a screenshot of the following statistics:

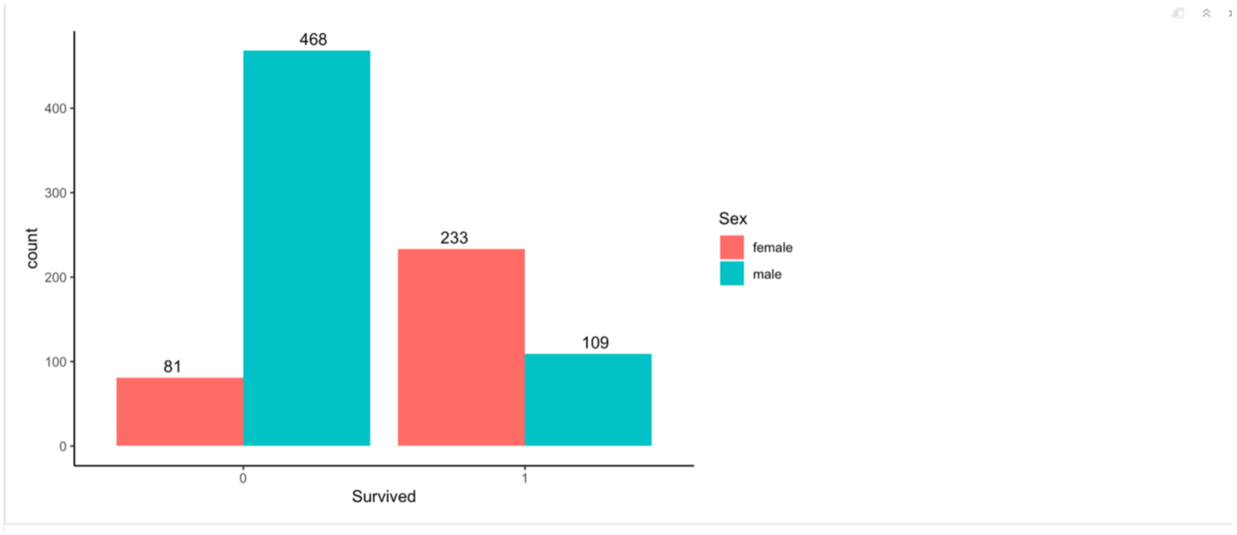


~~~
```{r}
summary(train)
```
~~~

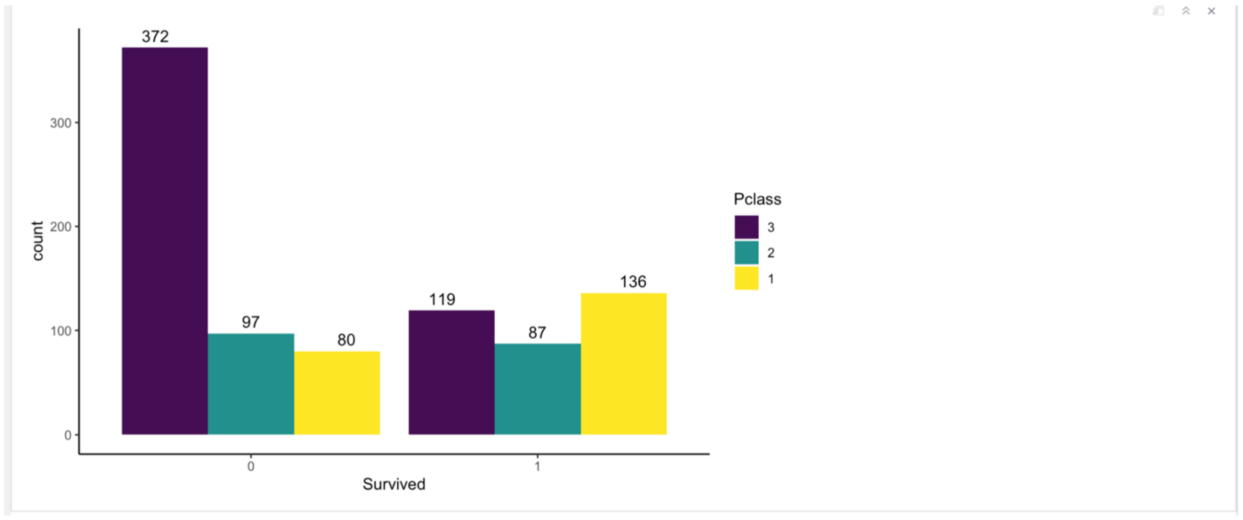
```
  PassengerId       Survived          Pclass          Name               Sex                 Age            SibSp           Parch       
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891         Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000  
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character   Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000  
 Median :446.0   Median :0.0000   Median :3.000   Mode  :character   Mode  :character   Median :28.00   Median :0.000   Median :0.0000  
 Mean   :446.0   Mean   :0.3838   Mean   :2.309                                         Mean   :29.70   Mean   :0.523   Mean   :0.3816  
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000                                         3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000  
 Max.   :891.0   Max.   :1.0000   Max.   :3.000                                         Max.   :80.00   Max.   :8.000   Max.   :6.0000  
                                                                                        NA's   :177                                     
    Ticket               Fare           Cabin             Embarked        
 Length:891         Min.   :  0.00   Length:891         Length:891        
 Class :character   1st Qu.:  7.91   Class :character   Class :character  
 Mode  :character   Median : 14.45   Mode  :character   Mode  :character  
                    Mean   : 32.20                                        
                    3rd Qu.: 31.00                                        
                    Max.   :512.33                                        
```

```
#These are the appropriate statistics.
```

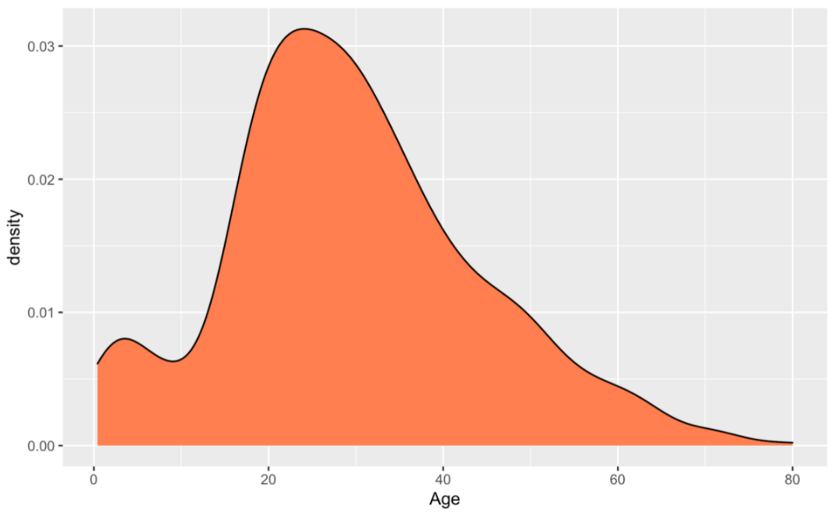
<u>Data Visualizations</u>



In the above chart, I've taken the survival count of passengers with respect to gender. The numbers 0 and 1 indicate whether the passengers didn't survive and survived respectively. From the graph, we can conclude that a smaller number of people survived, and, in those numbers, a larger proportion of females survived. I've taken this data because it gives us a sense of which gender had a better chance of survival.



Here, in the above graph, I’ve taken the data to find out the chance of survival of passengers according to which class they belong to. I feel that passenger class is an important factor to determine the outcome variable.
Again, 0 = Didn't Survive and 1 = Survived
We can infer that the chances of survival for passengers in 1st class was more than the others.



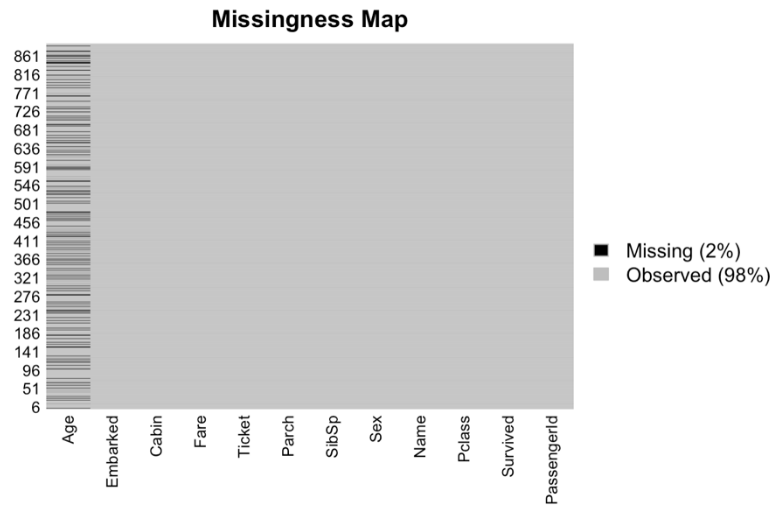
I've taken this graph to show which age group the passengers are from. Density plots give us this approximation. I feel that this gives us an idea as to what age group the passengers belong to and from here, we can get a bigger picture regarding the survival rate.
This graph shows us the age density and from it, we can conclude that most of the passengers were in the age group of 20 to 40.

Relationships I Observed

• **Pclass** We observe significant correlation (>0.5) among Pclass=1 and Survived.
• **Sex** - We confirm the observation during problem definition that Sex=female had very high survival rate at 74%
• **SibSp and Parch** These features have zero correlation for certain values.

Features Management/Data Cleaning

After going through the data, I found that there are a lot of missing values. On the first instinct, I found that the columns ‘Cabin’ and ‘Age’ has many NA values. So, I plotted a missingness map or a plot which shows the missing values. To plot the missingness map, I first loaded the ‘Amelia library’. Below is a screenshot of the R output:



As we can see, the ‘Cabin’ column has many NA values, and thus, we will drop it. Since the PassengerID is a unique identifier for the records, we will also drop it. I also realized that the Name, Fare, Embarked and Ticket columns will not decide the survival, so I dropped them as well. So I selected the remaining columns using the select() function:

```
library(dplyr)
data.frame = select(data.frame, Survived, Pclass, Age, Sex, SibSp,
Parch)
```

Next, to deal with the missing values, we drop the rows using the na.omit function as follows:

```
data.frame = na.omit(data.frame)
```

Next, I checked the structure of the data. To do this, we have to use the str() function:

```
'data.frame':	714 obs. of  6 variables:
 $ Survived: int  0 1 1 1 0 0 0 1 1 1 ...
 $ Pclass  : int  3 1 3 1 3 1 3 3 2 3 ...
 $ Age     : num  22 38 26 35 35 54 2 27 14 4 ...
 $ Sex     : chr  "male" "female" "female" "female" ...
 $ SibSp   : int  1 1 0 1 0 0 3 0 1 1 ...
 $ Parch   : int  0 0 0 0 0 0 1 2 0 1 ...
 - attr(*, "na.action")= 'omit' Named int [1:177] 6 18 20 27 29 30 32 33 37 43 ...
  ..- attr(*, "names")= chr [1:177] "6" "18" "20" "27" ...
```

Here, we can see that the Survived and Pclass column are integers. However, they are actually categorical variables. To convert them into categorical variables, we have to use the factor() function:

```
data.frame$Survived = factor(data.frame$Survived)

data.frame$Pclass = factor(data.frame$Pclass, order=TRUE, levels =
c(3, 2, 1))
```

Here, survived is a nominal categorical variable, whereas Pclass is an ordinal categorical variable. For an ordinal variable, we have to provide the order=TRUE and levels argument in the ascending order of the values (Pclass 3 < Pclass 2 < Pclass 1).

Learning Algorithm Training

Various Machine Learning Methods:

Decision Tree Model

Pros:

1) Data cleaning is easy: no need for data transformation (in linear regression, you need to transform the X variable if it has a nonlinear relationship with Y).
2) No need for variable selection, it is done automatically since it is part of the split selection.
3) Robust to outliers since the choice of a split depends on the ordering off values and not on the absolute magnitude of these values.
4) Non-parametric: there is no assumption on an articular relationship between the outcome and predictors.
5) nonlinear relationship: allows for a wide range of relationships between the predictors and the outcome variable

Cons:

1) sensitive to changes in the data
2) since the splits are done on one predictor at a time, rather than on a combination of predictors, the tree is likely to miss relationships between predictors
3) require large data sets

Logistic Regression Model

ADVANTAGES	DISADVANTAGES
Logistic regression is easier to implement, interpret, and very efficient to train.	If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
It makes no assumptions about distributions of classes in feature space.	It constructs linear boundaries.
It can easily extend to multiple classes(multinomial regression) and a natural probabilistic view of class predictions.	The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).	It can only be used to predict discrete functions. Hence, the dependent variable of Logistic Regression is bound to the discrete number set.
It is very fast at classifying unknown records.	Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.	Logistic Regression requires average or no multicollinearity between independent variables.
It can interpret model coefficients as indicators of feature importance.	It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.
Logistic regression is less inclined to over-fitting but it can overfit in high dimensional datasets.One may consider Regularization (L1 and L2) techniques to avoid over-fittingin these scenarios.	In Linear Regression independent and dependent variables are related linearly. But Logistic Regression needs that independent variables are linearly related to the log odds (log(p/(1-p)).

<u>Naïve Bayes Model</u>

Pros:

- The assumption that all features are independent makes naive bayes algorithm **very fast** compared to complicated algorithms. In some cases, speed is preferred over higher accuracy.
- It works well with high-dimensional data such as text classification, email spam detection.

Cons:

- The assumption that all features are independent is not usually the case in real life, so it makes naive bayes algorithm less accurate than complicated algorithms.

KNN Model

Pros:

1. No Training Period: KNN is called **Lazy Learner (Instance based learning)**. It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g., SVM, Linear Regression etc.
2. Since the KNN algorithm requires no training before making predictions, **new data can be added seamlessly** which will not impact the accuracy of the algorithm.
3. KNN is very **easy to implement**. There are only two parameters required to implement KNN i.e., the value of K and the distance function.

Cons:

1. Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing point is huge which degrades the performance of the algorithm.
2. Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.
3. Need feature scaling: We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.
4. Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

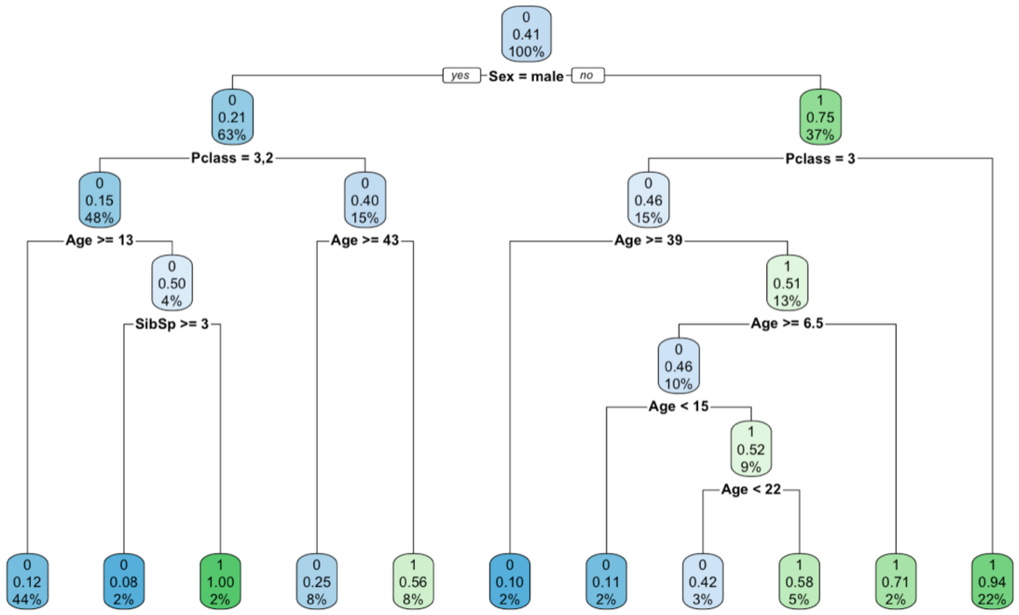
Implementing the Machine learning Methods:

Decision Tree Model

After using the following R code, I came to the following conclusions:

```
library(rpart)
library(rpart.plot)
fit <- rpart(Survived~., data = train, method = 'class')
rpart.plot(fit, extra = 106)
```

The decision tree model is available in the rpart library. The model is built using the rpart() function. The first parameter to this defines the target labels and the features. The attributes on the left of '~' specify the target label and attributes on left specify the features used for training. 'data' argument is your training data and method= 'class' tells that we are trying to solve a classification problem. The next function plots the decision tree as below.



Logistic Regression Model

For building a logistic regression model, we use the generalized linear model, glm() with the family= ‘binomial’ for classification.
We can perform scaling on the data using as.numeric() and scale() functions. The ‘.’ here specifies the complete dataset. It automatically ignores factors. We obtain predictions using the predict function with type = ‘response’ for obtaining the probabilities. In the table() function, we have passed an argument predict>0.68 which is a threshold that says, if the predicted probability is greater than 0.68, then we classify that record as 1 (which is survived).

```
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)
```

Furthermore, I got an accuracy of 80%.

Naïve Bayes Model

The Naïve Bayes Model is present in the e1071 library.

```
library(e1071)
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)
```

Furthermore, I got an accuracy of 81.81%.

KNN Model

The KNN model is available in the ‘class’ library. However, we should consider these things:

- knn() requires numeric variables. It throws error if you use factors in your data frame.
- knn() accepts only matrices or data frames as train and test arguments and not vectors.

We can use dummy() 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.
Note the ‘[,1]’ for train_labels and test_labels. This is because select() is returning a vector. But we need a data frame (or matrix).

```
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)
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)
```

Finally, after applying KNN to calculate the accuracy, I got an accuracy of 85.3%.

<u>Conclusion</u>

In this final project, I began by exploring the dataset, asking questions and finding answers through mining and visualization. I then dealt with missing values and replaced them appropriately, engineered a couple of new features and normalized the data. I also converted the categorical variables using dummy variables. Finally, I built a bunch of machine learning models and compared them to find the best model.

The Decision Tree model performed the best giving an accuracy of about 87%. Another algorithm based on decision trees is the Random Forest algorithm. Looking at the performance of decision trees, we can expect a similar or better performance using the ensemble method of Random Forest.

References

https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/
https://towardsdatascience.com/naive-bayes-classifier-explained-50f9723571ed