TITANIC

REPORT

The Columbian College of Art and Science
The George Washington University

ABSTRACT

The sinking of the Titanic which is the most serious peacetime shipwreck in modern history hit the world on April 15th, 1992. She hit an iceberg and caused a terrible disaster. At least 1500 people lost their lives among around 2224 passengers and crew abroad ⋯

Tan, Yueyan Zhu, Zhengting Hu, Jinyu

Machine Learning 1

Contents

Introduction

The sinking of the Titanic which is the most serious peacetime shipwreck in modern history hit the world on April 15th, 1992. She hit an iceberg and caused a terrible disaster. At least 1500 people lost their lives among around 2224 passengers and crew abroad.

The reason why Titanic sinking remains a mystery. Although scientists are focusing on exploiting the reason over several decades, what we figure from the dataset shows that some groups with typical features were more likely to survive. Thus, we want to dig the phenomenon deeper and explore the mystery in the data world.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

wen our project, we found out the connection between passengers' features and the probability of death. We used decision tree for finding out the most important features that affect a passenger's death in a shipwreck. Secondly, SVM algorism were used to find a specific decision boundary to classify and finally, we compare the performances of prediction by using these three algorisms.

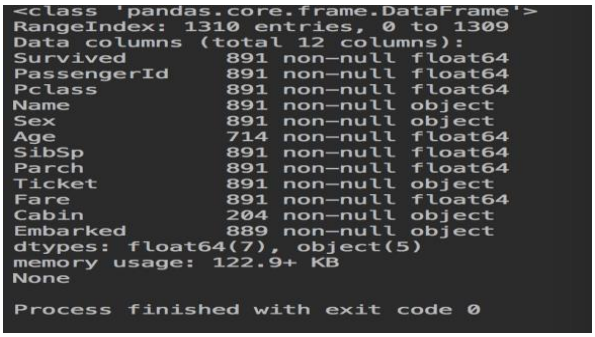
Description of the data set

Data Dictionary

wenformation of variables shows below:

Variable	Definition	Key
Survival	Survival	0=No; 1=Yes
Pclass	Ticket Class	1=1st (Upper); 2=2nd (Middle); 3=3rd (Lower)
Sex	Sex	
Age	Age in years	
Sibsp	# of siblings / spouses aboard the Titanic	Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)
Parch	# of parents / children aboard the Titanic	Parent = mother, father Child = daughter, son, stepdaughter, stepson Some children travelled only with a nanny, therefore parch=0 for them.
Ticket	Ticket Number	
Fare	Passenger fare	
Cabin	Cabin Number	
Embarked	Port of Embarkation	C= Cherbourg, Q= Queenstown, S=Southampton

After we import the Dataset into Python, we obtain the data feature shows below:



Findings

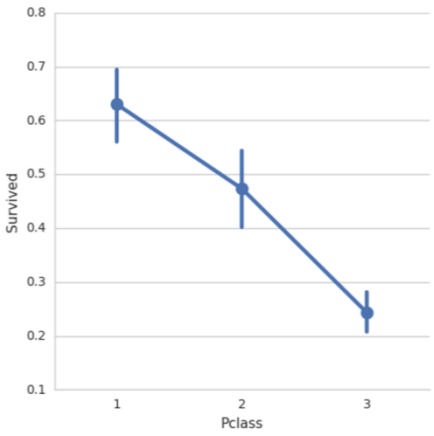
1. There are a total of 891 passengers in our training set.
2. The Age feature is missing approximately 19.8% of its values.
3. The Cabin feature is missing approximately 77.1% of its values.
4. The Embarked feature is missing 0.22% of its values, which is harmless.

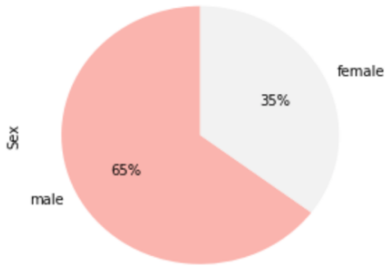
Data Analyzing:

We analyze the relationship between features and survival rates to show the critical factor of survival.

Pclass:

wet's easy to easy from the graph that the upper level class customers win higher survival rates in the sever shipwreck.

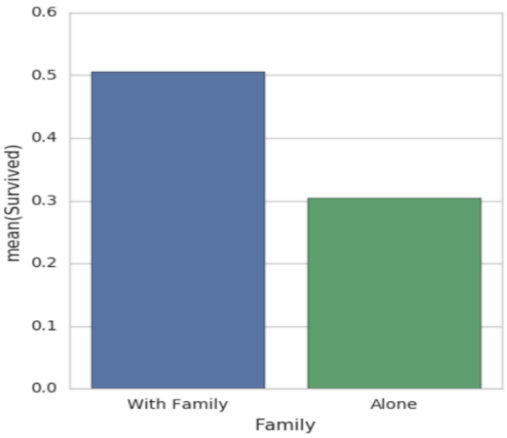


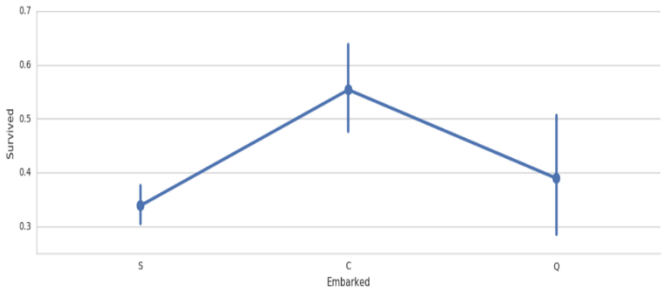


Sex:

From the pine chart, it shows that female has a high probability to survive in the disaster. (The percent represents the death rate)

Sibsp and Parch:





Combing the variables of Sibsp and Parch, we can obtain a new feature called Family/alone. From the graph below, we can see that the survival rate is higher if the passengers are on board with family.

Embarked:

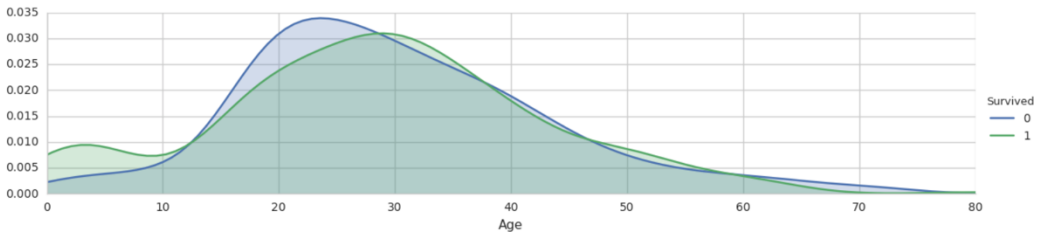
The graph shows that passengers embarked in Cherbourg are more likely to survive.

Fare:

This time we range the fare into serval group and the result shows that although it's not obvious, Passengers who bought higher fare has a lightly higher possibility to survive.

Fare	Survive
[0, 7.91]	0.197309
(7.91, 14.454]	0.303571
(14.454, 31]	0.454955
(31, 512.329]	0.581081

Age：



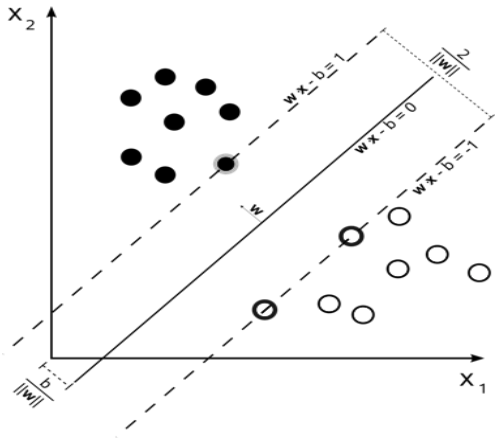
From the graph we get, we can get the phenomenon that young people get lots of chance to survive comparing with the child and old people.

Description of Algorithms

Support Vector Machines (SVM):

wen machine learning field, support vector machines (SVM) are supervised learning models that analyzing data for classification and regression.

A Support Vector Machine is a discriminative classifier formally defined by a separating hyperplane. wen other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. The easiest model picture looks like this:

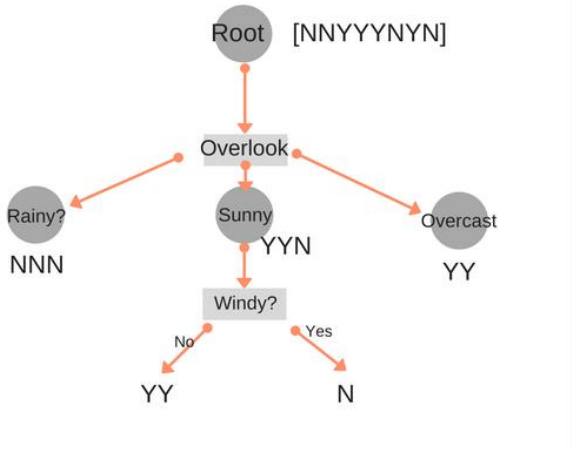


Decision Tree

wen Data Science field, a decision tree is support tool that uses a tree-like graph to show the decisions. A decision tree specially starts with a single node, and then branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch into other

possibilities. wet is one approach to display an algorithm that only contains conditional control statements in classification analysis.

One possible model looks like this



Some important factors of Decisions Tree are:

$$\text{Entropy} = \sum -pilog2pi$$

$$\text{Gini} = 1 - \sum (p_i^2)$$

wenformation Gain (weG) = Entropy (parent) – Average Entropy (children)

Naïve Bayes

wen machine learning field, naive Bayes classifiers are a group of simple probabilistic classifiers based on Bayes' theorem with independence assumptions between the features. wet is a powerful algorithm used for real time prediction, text classification, recommendation system, etc.

Key mathematic equation list below:

$$P(C(Class)|X(Features)) = \frac{P(X|C) \times P(C)}{P(X)}$$

P(C(Class)|X(Features)): Posterior Probability of class (C) given predictor (X)

P(X|C): Likelihood - the conditional probability of the predictor

P(C): Prior Probability of the Class

P(X): Total probability

Experimental Set up

As for experimental setup, we imported data from ‘data.csv’ and name it tt(test-data).

First, we cleaned data such as replacing missing characters as null, replacing categorical data with the most frequent value in that column and dropping “Cabin”, “ticket“and "name of passenger“. After cleaning, since the data type in each column is different, we use encoder to transform the data. And since we need to use SVM to train and predict the data, we need to normalize the data in advance, which will also make it faster to run. Secondly, we choose Survived column as target and all other columns are features. we split the data into train and test and choose 30% data as train data randomly. Next, we use three different algorithms to train the data and make predictions and for each algorithms we use classification report, accuracy score and confusion matrix to judge performance.

```
#decision tree
# perform training with giniIndex.
# creating the classifier object
clf_gini = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=3, min_samples_leaf=5)

# performing training
clf_gini.fit(X_train, y_train)

# perform training with entropy.
# Decision tree with entropy
clf_entropy = DecisionTreeClassifier(criterion="entropy", random_state=100, max_depth=3, min_samples_leaf=5)
clf_entropy.fit(X_train, y_train)
```

```
# display decision tree
dot_data = tree.export_graphviz(clf_gini, filled=True, rounded=True, class_names='survived', feature_names=tt.iloc[:, 0:].columns, out_file=None)

graph = graph_from_dot_data(dot_data)
graph.write_pdf("decision_tree_gini.pdf")
webbrowser.open_new(r'decision_tree_gini.pdf')

dot_data = tree.export_graphviz(clf_entropy, filled=True, rounded=True, class_names='survived', feature_names=tt.iloc[:, 0:].columns, out_file=
graph = graph_from_dot_data(dot_data)
graph.write_pdf("decision_tree_entropy.pdf")
webbrowser.open_new(r'decision_tree_entropy.pdf')
#%%
```

First is about the decision tree. we use two classifiers. One is gini and the other is entropy. we use code1to train the data separately and then we use code2 to predict y based on X_teat separately. Then we use code3to calculate metrics gini model and metrics entropy model. For clarity we use code4 to graph the confusion metric for gini model and entropy model. And from these two results we can know which classier can train better and which accuracy is better.

```
#prediction
y_pred_gini = clf_gini.predict(X_test)

y_pred_entropy = clf_entropy.predict(X_test)
```

```
# calculate metrics gini model
print("\n")
print("Results Using Gini Index: \n")
print("Classification Report: ")
print(classification_report(y_test,y_pred_gini))
print("\n")
print("Accuracy : ", accuracy_score(y_test, y_pred_gini) * 100)
print("\n")
print('-'*80 + '\n')

# calculate metrics entropy model
print("\n")
print("Results Using Entropy: \n")
print("Classification Report: ")
print(classification_report(y_test,y_pred_entropy))
print("\n")
print("Accuracy : ", accuracy_score(y_test, y_pred_entropy) * 100)
print('-'*80 + '\n')
```

Finally we plot decision tree using gini model and entropy model. From the graph, we can know exactly which feature is more important and which feature does not matter any more

```
# confusion matrix for gini model
conf_matrix = confusion_matrix(y_test, y_pred_gini)
class_names = tt.Survived.unique()
df_cm = pd.DataFrame(conf_matrix, index=class_names, columns=class_names)

plt.figure(figsize=(5,5))
hm = sns.heatmap(df_cm, cbar=False, annot=True, square=True, fmt='d', annot_kws={'size': 20}, yticklabels=df_cm.columns, xticklabels=df_cm.columns)
hm.yaxis.set_ticklabels(hm.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
hm.xaxis.set_ticklabels(hm.xaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
plt.ylabel('True label',fontsize=20)
plt.xlabel('Predicted label',fontsize=20)
plt.tight_layout()
plt.show()

# confusion matrix for entropy model
conf_matrix = confusion_matrix(y_test, y_pred_entropy)
class_names = tt.Survived.unique()
df_cm = pd.DataFrame(conf_matrix, index=class_names, columns=class_names)

plt.figure(figsize=(5,5))
hm = sns.heatmap(df_cm, cbar=False, annot=True, square=True, fmt='d', annot_kws={'size': 20}, yticklabels=df_cm.columns, xticklabels=df_cm.columns)
hm.yaxis.set_ticklabels(hm.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
hm.xaxis.set_ticklabels(hm.xaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
plt.ylabel('True label',fontsize=20)
plt.xlabel('Predicted label',fontsize=20)
plt.tight_layout()
plt.show()
```

Next is about SVM

```
clf = SVC(kernel="linear")
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

when code 1, we use to linear SVC to train the data and then we predict y based on X_teat. wen code 2, we calculate classification report and accuracy for later comparison. For clarity we use code4 to graph the confusion metric. Finally, we graph receiver operating characteristic graph.

```
# calculate metrics
print("\n")
print("Classification Report: ")
print(classification_report(y_test,y_pred))
print("\n")
print("Accuracy : ", accuracy_score(y_test, y_pred) * 100)
print("\n")
```

Next is about naive bayes.

```
clf = GaussianNB()

# performing training
clf.fit(X_train, y_train)
```

```
# make predictions

# predicton on test
y_pred = clf.predict(X_test)

y_pred_score = clf.predict_proba(X_test)
print(y_test)
print(y_pred_score)
```

wen code 1, we use to GaussianNB to train the data and then we predict y based on X_teat. wen code 2, we calculate classification report and accuracy for later comparison. For clarity we use code4 to graph the confusion metric

```
# confusion matrix

conf_matrix = confusion_matrix(y_test, y_pred)
class_names = tt['Survived'].unique()


df_cm = pd.DataFrame(conf_matrix, index=class_names, columns=class_names)

plt.figure(figsize=(5,5))
hm = sns.heatmap(df_cm, cbar=False, annot=True, square=True, fmt='d', annot_kws={'size': 20}, yticklabels=df_cm.columns, xticklabels=df_cm.colu
hm.yaxis.set_ticklabels(hm.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
hm.xaxis.set_ticklabels(hm.xaxis.get_ticklabels(), rotation=0, ha='right', fontsize=20)
plt.ylabel('True label',fontsize=20)
plt.xlabel('Predicted label',fontsize=20)
# Show heat map
plt.tight_layout()
plt.show()
```

```
# calculate metrics

print("\n")

print("Classification Report: ")
print(classification_report(y_test,y_pred))
print("\n")


print("Accuracy : ", accuracy_score(y_test, y_pred) * 100)
print("\n")

print("ROC_AUC : ", roc_auc_score(y_test,y_pred_score[:,1]) * 100)
print("\n")
```

Results

Decision Tree

For the first part of our project, we used Decision-Tree algorism for figuring out the degree of importance of each feature. We began with Decision Tree for the following reasons:

1. It is an algorism with widely used and could be well-visualized, which would be much easier for the readers from different background to understand our project results.
2. Decision Tree is simple to understand and to interpret, which also benefit the readers.
3. Decision Tree requires little data preparation. Other techniques such as SVM often require data normalization, dummy variables need to be created and blank values to be removed.
4. It is able to handle both numerical and categorical data. Other techniques like logistic regression and other statistical algorisms are usually specialized in analyzing datasets that have only one type of variable and the data we used (The Titanic data set) includes both categorical as well as numerical data, which means decision tree would be a good tool for the project.
5. It is possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model, such that we could fit the ROC for the prediction which would be much simple for readers to get our results.

Firstly, we got the decision tree by using entropy for calculating, as is shown in figure1:

Firstly, we got the decision tree by using entropy for calculating, as is shown in figure1:

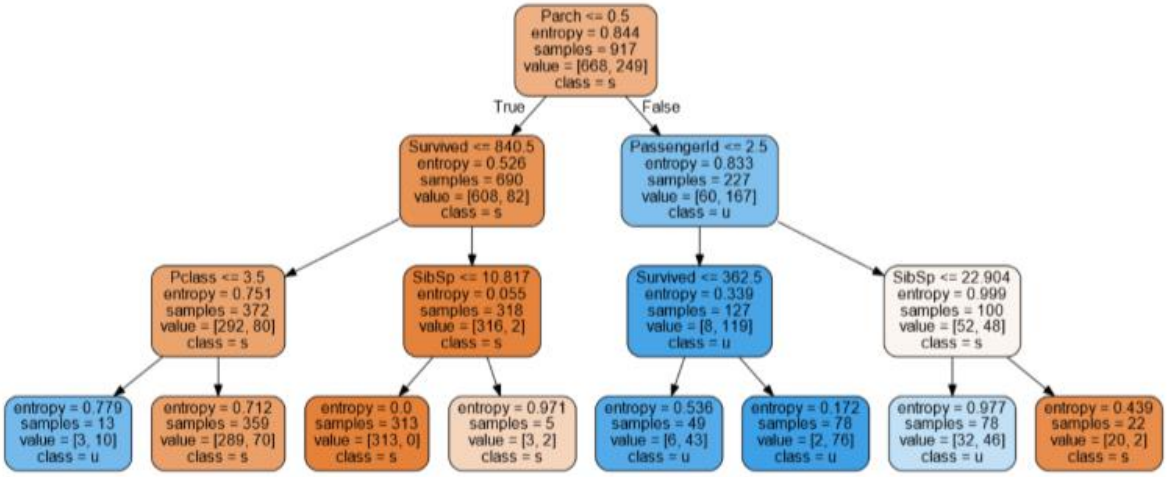


Figure 1 Decision Tree-Entropy

Before getting the result, we may guess that the most important features for affecting the possibility of survival would be gender or age, while according to the result, we could find out that the most important feature for deciding whether the passenger could survival from the shipwreck is “parch” (number of parents / children aboard the Titanic), which indicates that passengers with parents or children together would have much stronger desire of survival or higher rate or survival than others when facing the big disaster.

Moving down, we could see that “sibsp” (number of siblings / spouses aboard the Titanic) and “pclass” (Ticket class) are also important features that have effect on possibility of the survival. The reason of “sibsp” maybe the same with the feature “parch”. While “Pclass” also have such importance may because the rich or upper class always have priority over resources which is a harsh reality in our society.

Then, we got the decision tree by using Gini for calculating, as is shown in figure2:

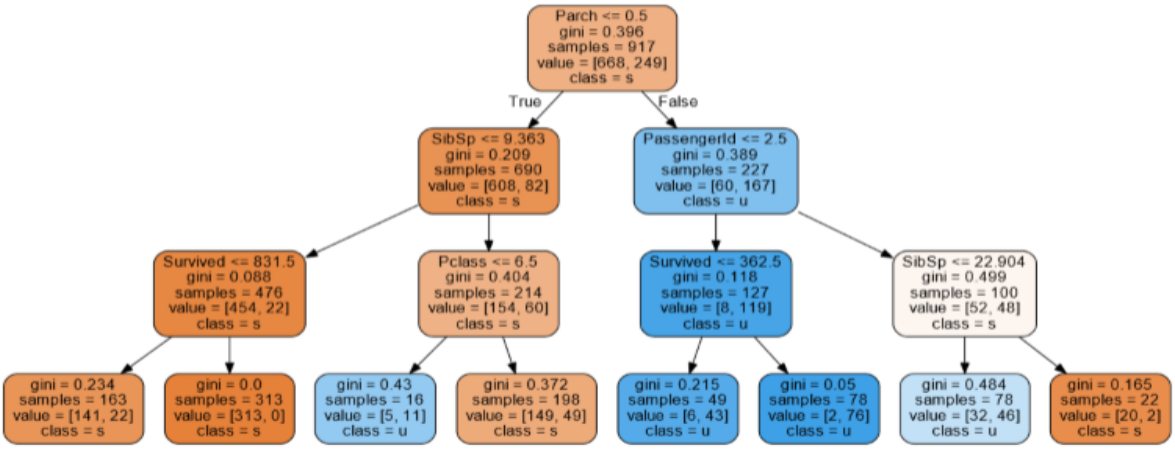


Figure 2 Decision Tree-GINI

Additionally, we also predict in both methods as is shown in Figure 3 and Figure 4:

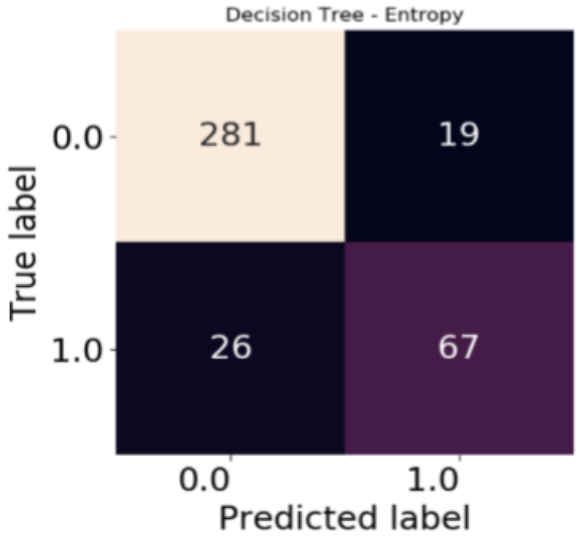


Figure 3 Entropy

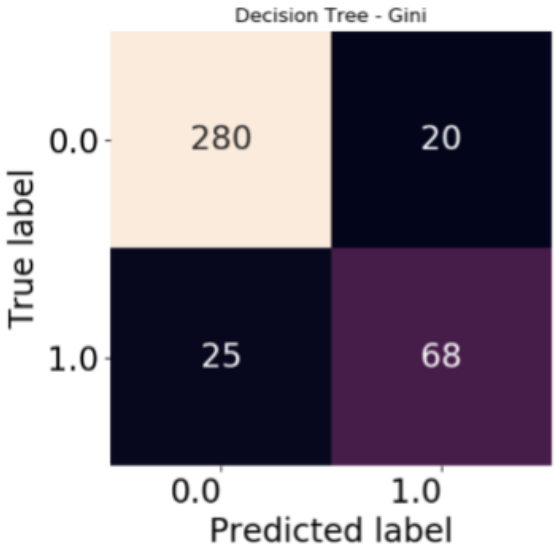


Figure 4 GINI

From the Figure 2, Figure 3 and Figure 4, it is easy to find out that Figure 2 is almost the same as figure 1 we got from entropy which means that both of these two calculations provide quite similar information and would not have much differences in the results. And for the performance of prediction with this two method, they were also the same (The prediction accuracy of both GINI and Entropy are 88.55%). While in the real life's application, more researchers would like to use Gini for calculation, then what is the differences between these two calculations ? After summarizing, we find that they are different in the following ways:

1. Entropy mostly used on categorical data set (discrete data sets).
2. Gini mostly used on contagious data set.
3. Entropy is slower to compute, while Gini is faster.

Support Vector Machine

For the second part of our project, Support Vector Machine were used for getting the specific decision boundary between two classes (survived and died), which may be helpful for discovering the effective features for that possibility of survival in shipwreck.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The reason we used SVM in this part because of the following reasons:

1. It has a regularization parameter, which makes the user think about avoiding over-fitting. Secondly it uses the kernel trick, so you can build in expert knowledge about the problem via engineering the kernel.
2. An SVM is defined by a convex optimization problem (no local minima) for which there are efficient methods (e.g. SMO).
3. It is an approximation to a bound on the test error rate, and there is a substantial body of theory behind it which suggests it should be a good idea.

In this part of our project, the specific decision boundary would be helpful for classify correctly and we could get the equation of the decision boundary. Besides, SVM also used for predict the possibility of survival of a passenger in the shipwreck, therefore the results of SVM and Decision Tree could be used for compare the performance of different algorisms.

According to the results we got from SVM, as shown in Figure 5 and Figure 6:

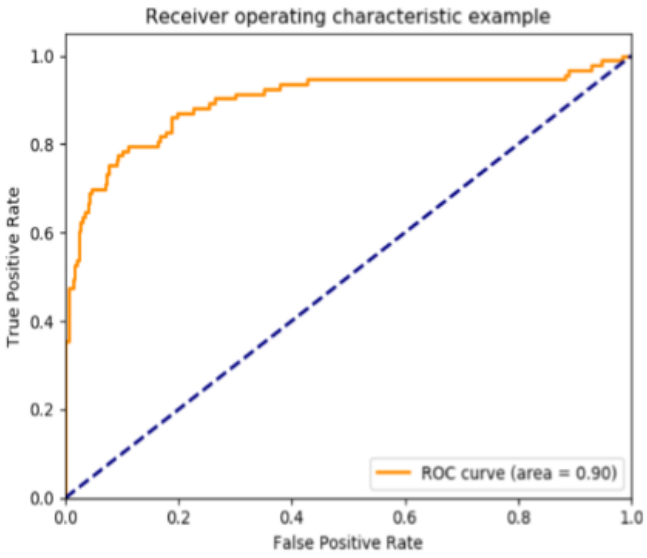


Figure 5 ROC Curve

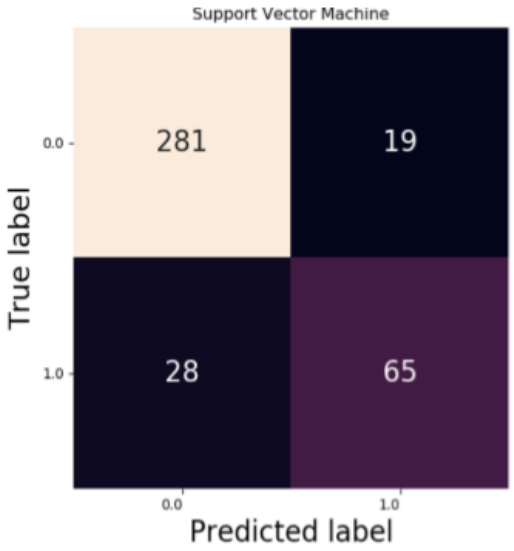


Figure 6 SVM

According to the Figure 5, the AUC=0.9 which means SVM is a good classifier in this case. The AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example. ROC Curve were used in this part is because when the distribution of positive and negative samples in the test set changes, the ROC curves can remain unchanged. Class imbalance is often found in the actual data set, that is, the negative sample is much more than the positive sample (or the opposite), and the distribution of the positive and negative samples in the test data may change over time.

From Figure 6, we could indicate that the prediction accuracy of SVM is the same as Decision Tree (Entropy), which is about 87.78%, therefore, we may draw the conclusion that there is no significant difference between the performances of Decision Tree and SVM algorism in predicting the possibility of survival in shipwreck (Titanic case).

Naïve Bayes

For the last part of research, we intended to find out that whether Naïve Bayes algorism would perform better in prediction than Decision Tree or SVM, Since Naïve Bayes has lots of advantages than other Machine Learning algorisms:

1. NB is a very simple algorism which is easy to implement and fast.
2. If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression.
3. Even if the NB assumption doesn't hold, it works great in practice.
4. NB only needs less training data.
5. NB is highly scalable. It scales linearly with the number of predictors and data points.
6. It can be used for both binary and multi-class classification problems.
7. It can make probabilistic predictions.
8. It can handle continuous and discrete data.
9. It is not sensitive to irrelevant features.

From the results we got from SVM, as shown in Figure 7:

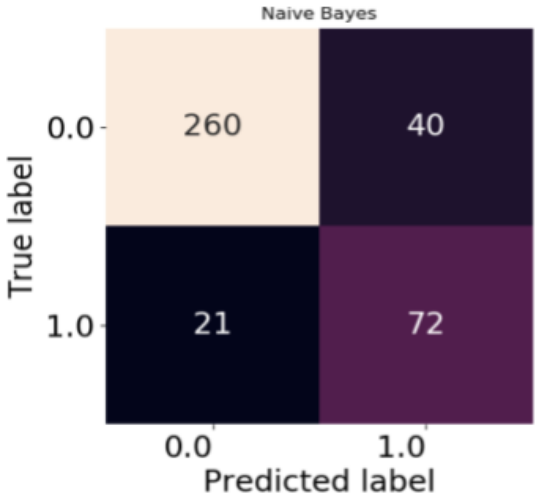


Figure 7 Naïve Bayes

According to the results we got from python in Figure 7, we could find out that the false positive rate is much higher than the other two algorisms (accuracy of prediction is only 84.47%), which means in real life of future, rescue workers would spend much time for recuing trapped person

who may has died in the shipwreck, and it would absolutely waste much rescuing time and money if researchers doing such prediction by using Naïve Bayes algorism. Taking all these above into consideration, we may draw the conclusion that although Naïve Bayes has lots of advantages, it is not a appropriate algorism for prediction in this case.

Summary and conclusions

All in all, compared three algorithms, we know decision tree and support vector machine are more accurate than naive bayes based on metrics accuracy report. And the most important result is that the most important features that impact passengers' survival rates are parch(number of parents/children aboard the Titanic), sibsp(number of siblings/spouses aboard the Titanic) and pclass(ticket class).

From the project, we learned to preprocess data. Since we use decision tree, support vector machine and naive bayes, we need to delete columns that we do not need, to replace missing characters as NaN, to replace categorical mushroom_data with the most frequent value in that column and finally to normalize the data. Secondly, we learned to run three algorithms and to use accuracy, classification report and confusion metrics to compare which algorithm is more accurate. And there are still a lot of things we can prove. wef we have more time we will define a function to use naive bayes to predict a passenger's survival rate based on the passenger's features.

References

[1] Sapatinas, T. (n.d.). The Elements of Statistical Learning. Journal of the Royal Statistical Society: Series A (Statistics in Society). Oxford, UK: Blackwell Publishing. doi:10.1111/j.1467-985X.2004.298_11.x

[2] Durbin, R., Miall, C., & Mitchison, G. (1989). The Computing neuron . Wokingham, England ;: Addison-Wesley Pub. Company.

[3] Raschka, S. (2015). Python Machine Learning. Birmingham: Packt Publishing Ltd.

[4].https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy/notebook

[5] https://www.kaggle.com/c/titanic