Summer Training Report

On

Titanic Survival Prediction

Submitted

By

Ayush Pandey

Roll No: 171500077

Department of Computer Engineering &

Application

GLA University

Mathura-281406, India

2019

Department of Computer Engineering and Applications GLA University, Mathura 17km, Stone NH#2, Mathura-Delhi Road, P.O.–Chaumuha, Mathura-281406

Declaration

I hereby declare that the work which is being presented in Summer Training "Titanic Survival Prediction", in partial fulfillment of the requirements for summer training viva voce, is an authentic record of my own work carried under the supervision of "E & ICT Academy, IIT Kanpur".

Signature of candidate:

Name of candidate: Ayush Pandey

Roll No.: 171500077

Course: B.Tech(CSE)

Year: 3rd

Semester: 5th

Summer Training Synopsis
B.Tech.(CSE)-batch 2017-2021

Student Information:

Name: Ayush Pandey	University roll no.: 171500077
Mobile: 9653067982	Email: ayush.pandey_cs17@gla.ac.in

Information about Industry/Organization:

Organization name with full address:	E & ICT Academy, IIT Kanpur, Kalyanpur, Uttar Pradesh-208016
Contact person:	Dr. B.V. Phani (chief investigator officer ICT Academy IITK) Mobile: 7992088885

Project Information:

Title of project	Titanic Survival Prediction
Role & responsibility	Learner
Technical details	Hardware requirements: laptop Software requirements: python3, pycharm IDLE & different python libraries
Training implementation details	Fully implemented
Training period	Start date: 12/06/2019 End date: 10/07/2019 Duration of training: 4 weeks

Summary

The available dataset for this project is provided by Kaggle, which is a platform for predictive models development competitions in which researchers post data as a competition. Then users compete to produce the best models for predicting and describing the data by submitting a separate unpredicted dataset where competitors apply their predictive models to guess survival for each person in the latter dataset.

This dataset contains information about a group of Passengers, about 891 folks that have been provided by kaggle. The data is available in comma separated format , each tuple has 12 features to represent a single passenger; which are ticket number, ticket class (class 1, class 2, class 3) that represents in which class the passenger was set, the higher the class the more luxurious the class was and the closer to the lifeboats, passenger name that is consists of the title (Mr, Ms, Mrs, Master, Miss, Don, etc), age in years, gender, family information about how many children and spouse this passenger has so if the passenger has 2 children and a spouse then this field value is 3, embarking port represents the city of embarking which are Cherbourg in France and Queenstown and Southampton in the UK , fare paid per person and the cabin of settlement and finally the survived field that is already provided the truth whether everybody if survived (1) or died (0).

Acknowledgement

In the present world of competition there is a race of existence in which those are having will to come forward succeed. Project is like a bridge between theoretical and practical working. With this willing I joined this particular project. First of all, I would like to thank the supreme power the almighty God who is obviously the one has always guided me to work on the right path of life. Without his grace this project could not become a reality. Next to him are my parents, whom I am greatly indebted for me brought up with love and encouragement to this stage.

I am feeling oblige in taking the opportunity to sincerely thanks to Dr. B. V. Phani (chief investigator officer , E & ICT Academy, IITK) and special thanks to my worthy trainer of machine learning Jeetendra Singh. Moreover , I am highly obliged in taking the opportunity to sincerely thanks to all the supporting staff member of ICT Academy for their generous attitude and friendly behavior. At last but not the least I am thankful to my trainer and friends who have been always helping me and encouraging me though the training duration. I have no valuable words to express my thanks, but my heart is still full of the favor received from every person.

Abstract

Technology advances more and more every day, and the collected data is growing day after day and becomes extremely valuable and these days, data is the fuel of future (Economist, 2017) and we can gain much more useful information as data science tools evolve. In this project, we will focus on The Titanic Sink historical event, which is still one of the most memorable occurrences from that time in the early 1910s. We will focus on the final status of the passengers by proposing different angles of visualization for the relations between features. Also by developing a predictive model using data science techniques such as Random Forests, Decision Trees and K-Nearest-Neighbors (K-NN) to predict whether each person survived or perished, based on the data available from that time to gain as much knowledge to produce better effective decisions related to this event.

Contents

CHAPTER-1 INTRODUCTION

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

1.0 Introduction

The goal of the project was to predict the survival of passengers based off a set of data. We used Kaggle competition "Titanic Survival Prediction" to retrieve necessary data and evaluate accuracy of our predictions. The historical data has been split into two groups, a 'training set' and a 'test set'. For the training set, we are provided with the outcome (whether or not a passenger survived). We used this set to build our model to generate predictions for the test set.

For each passenger in the test set, we had to predict whether or not they survived the sinking. Our score was the percentage of correctly predictions.

In our work, we learned

- Programming language Python and its libraries NumPy (to perform matrix operations) and SciKit-Learn (to apply machine learning algorithms)
- Several machine learning algorithms (decision tree, random forests, extra trees, linear regression)
- Feature Engineering techniques

We use

- Pycharm integrated development environment
- Python 3.7.3 with the libraries numpy, sklearn, an matplotlib
- Microsoft Excel

1.1 Work Plan

1. Learn programming language Python
2. Learn Shennon Entropy and write Python code to compute Shennon Entropy
3. Get familiar with Kaggle project and try using Pivot Tables in Microsoft Excel to analyze the data.
4. Learn to use SciKit-Learn library in Python, including
    a. Building decision tree
    b. Building Random Forests
    c. Building Extra Trees
    d. Using Linear Regression algorithm
5. Performing Feature Engineering, applying machine learning algorithms, and analyzing results

1.2 Training and Test Data

Training and Test data come in CSV file and contain the following fields:

- Passenger ID
- Passenger Class
- Name
- Sex
- Age

- Number of passenger's siblings and spouses on board
- Number of passenger's parents and children on board
- Ticket
- Fare
- Cabin
- City where passenger embarked

CHAPTER-2 FEATURE ENGINEERING

2.0 Feature Engineering

Since the data can have missing fields, incomplete fields, or fields containing hidden information, a crucial step in building any prediction system is Feature Engineering. For instance, the fields Age, Fare, and Embarked in the training and test data, had missing values that had to be filled in the field Name while being useless itself, contained passenger's Title (Mr., Mrs., etc.), we also used passenger's surname to distinguish families on board of Titanic. Below is the list of all changes that has been made to the data.

2.1 Extracting Title from Name

The field Name in the training and test data has the form "Braund, Mr. Owen Harris". Since name is unique for each passenger, it is not useful for our prediction system. However, a passenger's title can be extracted from his or her name. We found 10 titles:

Index	Title	Number of occurrences
0	Col.	4
1	Dr..	8
2	Lady	4

3	Master	61
4	Miss	262
5	Mr	757
6	Mrs	198
7	Ms	2
8	Rev	8
9	Sir	5

Table-2.1 Title with their corresponding no. of occurrence

We can see that title may indicate passenger's sex (Mr. vs Mrs.), class (Lady vs Mrs.), age (Master vs Mr.), profession (Col., Dr., and Rev.).

2.2 Calculating Family Size

It seems advantageous to calculate family size as follows

Family_Size = Parents_Children +Siblings_Spouses+1

2.3 Extracting Deck from Cabin

The field Cabin in the training and test data has the form "C85", "C125", where C refers to the deck label. We found 8 deck labels: A, B, C, D, E, F, G, T. We see deck label as a refinement of the passenger's class field since the decks A and B were intended for passengers of the first class, etc.

2.4 Extracting Ticket_Code from Ticket

The field Ticket in the training and test data has the form "A/5 21171". Although we couldn't understand meaning of letters in front of numbers in the field Ticket, we extracted those letters and used them in our prediction system. We found the following letters

Index	Ticket Code	Number of occurrences
0	No Code	961
1	A	42
2	C	77
3	F	13
4	L	1
5	P	98
6	S	98
7	W	19

Table 2.2 Ticket code with corresponding no. of occurrence

2.5 Filling in missing values in the fields Fare, Embarked, and Age

Since the number of missing values was small, we used median of all Fare values to fill in missing Fare fields, and the letter 'S' (most frequent value) for the field Embarked.

In the training and test data, there was significant amount of missing Ages. To fill in those, we used Linear Regression algorithm to predict Ages based on all other fields except Passenger_ID and Survived.

CHAPTER-3 IMPORTANCE OF FIELDS

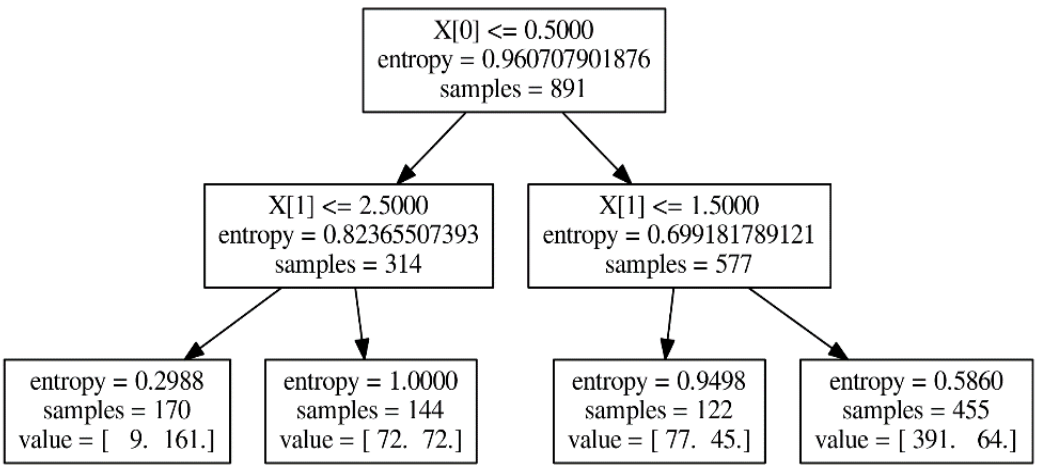
3.0 Importance of fields

Decision Trees algorithm in the library SciKit-Learn allows to evaluate importance of each field used for prediction.

We can say that the field Sex is the most important one for prediction, followed by Title, Fare, Age, Class, Deck, Family_Size, etc

3.1 Decision Trees

Our prediction system is based on growing Decision Trees to predict the survival status. A typical Decision Tree is pictured below



Dia.3.1 The survival status of person

The basic algorithm for growing Decision Tree:

1. Start at the root node as parent node
2. Split the parent node based on field $X[i]$ to minimize the sum of child nodes uncertainty (maximize information gain)
3. Assign training samples to new child nodes
4. Stop if leave nodes are pure or early stopping criteria is satisfied, otherwise repeat step 1 and 2 for each new child node

Stopping Rules:

1. The leaf nodes are pure
2. A maximal node depth is reached
3. Splitting a node does not lead to an information gain

In order to measure uncertainty and information gain, we used the formula

$IG(D_p) = I(D_p) - {}^{N}\text{left}/\ N_p\, I(D_{left}) - {}^{N}\text{right}\ /N_p\, I(D_{right})$

where

• IG: Information Gain

• I: Impurity (Uncertainty Measure)

• N_p, N_{left}, N_{right}: number of samples in the parent, the left child, and the right child nodes

• D_p, D_{left}, D_{right}: raining subset of the parent, the left child, and the right child nodes

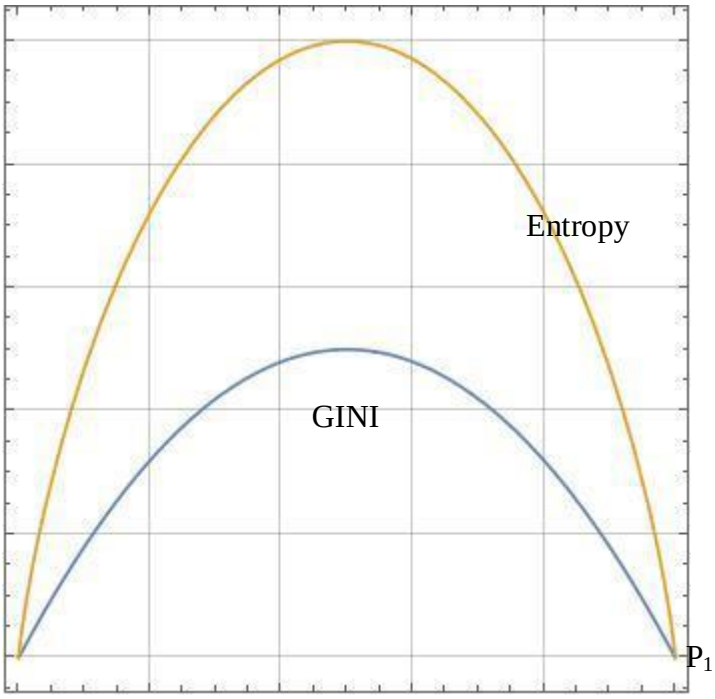
For Uncertainty Measure, we used Entropy defined by

$I(p_1,p_2) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$

and GINI index defined by

$I(p_1,p_2) = 2p_1p_2$

The graphs of both measures are given below



Diag.3.2 The relationship status of GINI index and Entropy wrt prob.

We can see on the graph that when probability of an event is 0 or 1, then the uncertainty measure equals to 0, while if probability of an event is close to 1⁄2, then the uncertainty measure is maximum.

3.2 Random Forest and Extra Trees

One common issue with all machine learning algorithms is Overfitting. For Decision Tree, it means growing too large tree (with strong bias, small variation) so it loses its ability to generalize the data and to predict the output. In order to deal with overfitting, we can grow several decision trees and take the average of their predictions. The library SciKit-Learn provides to such algorithm Random Forest and ExtraTrees.

In Random Forest, we grow *N* decision trees based on randomly selected subset of the data and randomly selected *M* fields, where M = √total # of fields.
In ExtraTrees, in addition to randomness of subsets of the data and of field, splits of nodes are chosen randomly.

CHAPTER-4 CONCLUSION

As a result of our work, we gained valuable experience of building prediction systems and achieved 80.383% of correct predictions.

- We performed featured engineering techniques
- Changed alphabetic values to numeric
- Calculated family size
- Extracted title from name and deck label from ticket number
- Used linear regression algorithm to fill in missing ages
- We used several prediction algorithms in python
- Decision tree
- Random forests
- Extra trees
- We achieved our best score 80.383% correct predictions

APPENDICES

Code Implementation:

```
import numpy as np
import pandas as pd
pd.set_option('display.width', 1000)
pd.set_option('display.max_column',
16)
pd.set_option('precision', 2)
import matplotlib.pyplot as plt
import seaborn as sbn
import warnings
warnings.filterwarnings('ignore')
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
print( train.describe() )
print( "\n" )
print( train.describe(include="all") )
print( "\n" )
print( "\n\n" , train.columns )
print()
print( train.head() )
print()
print( train.sample(5) )
print( "Data types for each feature : -
" )
print( train.dtypes )
print( train.describe(include = "all") )
print()
print( pd.isnull(train).sum() )
sbn.barplot(x="Sex", y="Survived",
data=train)
plt.show()
print( "----------\n\n" )
print( train )
print( "----------\n\n" )
print( train["Survived"] )
print( "----------\n\n" )
print( train["Sex"] == 'female' )
print( "**********\n\n" )
print( train["Survived"][
train["Sex"] == 'female' ] )
print( "**********\n\n" )
print(train["Survived"][train["Sex"]
== 'female'].value_counts() )
print(
"==========
==========\n\n" )
print( train["Survived"][train["Sex"]
== 'female'].value_counts(normalize =
True))
print( train["Survived"][train["Sex"]
== 'female'].value_counts(normalize =
True)[1] )
print( "Percentage of females who
survived:",
train["Survived"][train["Sex"] ==
```

```
'female'].value_counts(normalize =
True)[1]*100  )
print( "Percentage of males who
survived:",
train["Survived"][train["Sex"] ==
'male'].value_counts(normalize =
True)[1]*100  )
sbn.barplot(x="Pclass",
y="Survived", data=train)
plt.show()
print("Percentage of Pclass = 1 who
survived:",
train["Survived"][train["Pclass"] ==
1].value_counts(normalize =
True)[1]*100)
print("Percentage of Pclass = 2 who
survived:",
train["Survived"][train["Pclass"] ==
2].value_counts(normalize =
True)[1]*100)
print("Percentage of Pclass = 3 who
survived:",
train["Survived"][train["Pclass"] ==
3].value_counts(normalize =
True)[1]*100)
print()
print( "Percentage of Pclass = 1 who
survived:\n\n",
train["Survived"][train["Pclass"] ==
1].value_counts()  )
print()
print( "Percentage of Pclass = 1 who
survived:\n\n",
train["Survived"][train["Pclass"] ==
1].value_counts(normalize = True)    )


print()
print( "Percentage of Pclass = 1 who
survived:\n\n",
train["Survived"][train["Pclass"] ==
1].value_counts(normalize = True)[1]
)
sbn.barplot(x="SibSp",
y="Survived", data=train)
print("Percentage of SibSp = 0 who
survived:",
    train["Survived"][train["SibSp"]
== 0].value_counts(normalize =
True)[1]*100)
print("Percentage of SibSp = 1 who
survived:",
    train["Survived"][train["SibSp"]
== 1].value_counts(normalize =
True)[1]*100)
print("Percentage of SibSp = 2 who
survived:",
    train["Survived"][train["SibSp"]
== 2].value_counts(normalize =
True)[1]*100)
plt.show()
sbn.barplot(x="Parch",
y="Survived", data=train)
```

```
plt.show()
train["Age"] = train["Age"].fillna(-0.5)
test["Age"] = test["Age"].fillna(-0.5)
bins =  [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Senior']
train['AgeGroup'] = pd.cut(train["Age"], bins, labels = labels)
test['AgeGroup'] = pd.cut(test["Age"], bins, labels = labels)
print( train  )
sbn.barplot(x="AgeGroup", y="Survived", data=train)
plt.show()
train["CabinBool"] = (train["Cabin"].notnull().astype('int'))
test["CabinBool"] = (test["Cabin"].notnull().astype('int'))
print( "#################################\n\n"  )
print( train  )
print("Percentage of CabinBool = 1 who survived:", train["Survived"][train["CabinBool"] == 1].value_counts( normalize = True)[1]*100)
print("Percentage of CabinBool = 0 who survived:", train["Survived"][train["CabinBool"] == 0].value_counts( normalize = True)[1]*100)
sbn.barplot(x="CabinBool", y="Survived", data=train)
plt.show()
print( test.describe(include="all")  )
train = train.drop(['Cabin'], axis = 1)
test = test.drop( ['Cabin'], axis = 1)
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
print( "Number of people embarking in Southampton (S):" ,  )
print( "\n\nSHAPE = " , train[train["Embarked"] == "S"].shape  )
print( "SHAPE[0] = " , train[train["Embarked"] == "S"].shape[0]  )
southampton = train[train["Embarked"] == "S"].shape[0]
print( southampton  )
```

```
print( "Number of people embarking in Cherbourg (C):" , )
cherbourg = train[train["Embarked"] == "C"].shape[0]
print( cherbourg )
print( "Number of people embarking in Queenstown (Q):" , )
queenstown = train[train["Embarked"] == "Q"].shape[0]
print( queenstown )
train = train.fillna({"Embarked": "S"})
combine = [train, test]
print( combine[0] )
for dataset in combine:
    dataset['Title'] = dataset['Name'].str.extract(', ([A-Za-z]+)\.', expand=False)
print( "\n\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n" )
print( train )
print()
print( pd.crosstab(train['Title'], train['Sex'] )  )
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(
        ['Lady', 'Capt', 'Col','Don', 'Dr', 'Major', 'Rev', 'Jonkheer', 'Dona'],
        'Rare')
    dataset['Title'] = dataset['Title'].replace(['Countess', 'Sir'], 'Royal')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
print( "\n\nAfter grouping rare title : \n" , train )
print( train[['Title', 'Survived']].groupby(['Title'], as_index=True).count() )
print( "\nMap each of the title groups to a numerical value." )
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Royal": 5, "Rare": 6}
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)
print( "\n\nAfter replacing title with neumeric values.\n" )
print( train )
mr_age = train[train["Title"] ==
```

```
1]["AgeGroup"].mode()
print( "mode() of mr_age : ", mr_age
)
print( "\n\n" )
miss_age = train[train["Title"] ==
2]["AgeGroup"].mode()
print( "mode() of miss_age : ",
miss_age )
print( "\n\n" )
mrs_age = train[train["Title"] ==
3]["AgeGroup"].mode()
print( "mode() of mrs_age : ",
mrs_age )
print( "\n\n" )
master_age = train[train["Title"] ==
4]["AgeGroup"].mode()
print( "mode() of master_age : ",
master_age )
print( "\n\n" )
royal_age = train[train["Title"] ==
5]["AgeGroup"].mode()
print( "mode() of royal_age : ",
royal_age )
print( "\n\n" )
rare_age = train[train["Title"] ==
6]["AgeGroup"].mode()
print( "mode() of rare_age : ",
rare_age )
print(
"\n\n**************************
**********************\n\n" )
print( train.describe(include="all") )
print( train )
print( "\n\n********
train[AgeGroup][0] : \n\n" )
for x in range(10) :
    print( train["AgeGroup"][x] )
age_title_mapping = {1: "Young
Adult", 2: "Student",
                3: "Adult", 4: "Baby", 5:
"Adult", 6: "Adult"}
for x in
range(len(train["AgeGroup"])):
    if train["AgeGroup"][x] ==
"Unknown":
        train["AgeGroup"][x] =
age_title_mapping[ train["Title"][x]
]
for x in
range(len(test["AgeGroup"])):
    if test["AgeGroup"][x] ==
"Unknown":
        test["AgeGroup"][x] =
age_title_mapping[test["Title"][x]]
print( "\n\nAfter replacing Unknown
values from AgeGroup column : \n"
)
print( train )
age_mapping = {'Baby': 1, 'Child': 2,
'Teenager': 3,
            'Student': 4, 'Young Adult':
5,
```

```
            'Adult': 6, 'Senior': 7}
train['AgeGroup'] =
train['AgeGroup'].map(age_mapping)
test['AgeGroup'] =
test['AgeGroup'].map(age_mapping)
print()
print( train  )
train = train.drop(['Age'], axis=1)
test = test.drop(['Age'], axis=1)
print( "\n\nAge column droped."  )
print( train  )
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)
sex_mapping = {"male": 0, "female":
1}
train['Sex'] =
train['Sex'].map(sex_mapping)
test['Sex'] =
test['Sex'].map(sex_mapping)
print( train  )
embarked_mapping = {"S": 1, "C": 2,
"Q": 3}
train['Embarked'] =
train['Embarked'].map(embarked_ma
pping)
test['Embarked'] =
test['Embarked'].map(embarked_map
ping)
print()
print( train.head()  )
for x in range(len(test["Fare"])):
    if pd.isnull(test["Fare"][x]):
        pclass = test["Pclass"][x]
#Pclass = 3
        test["Fare"][x] = round(train[
train["Pclass"] == pclass
]["Fare"].mean(), 2)
train['FareBand'] =
pd.qcut(train['Fare'], 4,
                    labels = [1, 2, 3, 4])
test['FareBand'] =
pd.qcut(test['Fare'], 4,
                    labels = [1, 2, 3, 4])
train = train.drop(['Fare'], axis = 1)
test = test.drop(['Fare'], axis = 1)
print( "\n\nFare column droped\n"  )
print( train  )
print()
print( test.head()  )
from sklearn.model_selection import
train_test_split
input_predictors =
train.drop(['Survived',
'PassengerId'], axis=1)
ouptut_target = train["Survived"]

x_train, x_val, y_train,
y_val=train_test_split(
    input_predictors, ouptut_target,
test_size = 0.20, random_state = 7)
from sklearn.metrics import
accuracy_score
```

```
from sklearn.linear_model import
LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_val)
acc_logreg =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-1: Accuracy of
LogisticRegression : ", acc_logreg  )
from sklearn.naive_bayes import
GaussianNB
gaussian = GaussianNB()
gaussian.fit(x_train, y_train)
y_pred = gaussian.predict(x_val)
acc_gaussian =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-2: Accuracy of
GaussianNB : ", acc_gaussian  )
from sklearn.svm import SVC
svc = SVC()
svc.fit(x_train, y_train)
y_pred = svc.predict(x_val)
acc_svc =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-3: Accuracy of
Support Vector Machines : ",
acc_svc  )
from sklearn.svm import LinearSVC
linear_svc = LinearSVC()
linear_svc.fit(x_train, y_train)
y_pred = linear_svc.predict(x_val)
acc_linear_svc =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-4: Accuracy of
LinearSVC : ",acc_linear_svc  )
from sklearn.linear_model import
Perceptron
perceptron = Perceptron()
perceptron.fit(x_train, y_train)
y_pred = perceptron.predict(x_val)
acc_perceptron =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-5: Accuracy of
Perceptron : ",acc_perceptron  )
from sklearn.tree import
DecisionTreeClassifier
decisiontree = DecisionTreeClassifier()
decisiontree.fit(x_train, y_train)
y_pred = decisiontree.predict(x_val)
acc_decisiontree =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-6: Accuracy of
DecisionTreeClassifier : ",
acc_decisiontree  )
from sklearn.ensemble import
RandomForestClassifier
```

```
randomforest =
RandomForestClassifier()
randomforest.fit(x_train, y_train)
y_pred = randomforest.predict(x_val)
acc_randomforest =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-7: Accuracy of
RandomForestClassifier :
",acc_randomforest  )
from sklearn.neighbors import
KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred = knn.predict(x_val)
acc_knn =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-8: Accuracy of k-
Nearest Neighbors : ",acc_knn  )
from sklearn.linear_model import
SGDClassifier
sgd = SGDClassifier()
sgd.fit(x_train, y_train)
y_pred = sgd.predict(x_val)
acc_sgd =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-9: Accuracy of
Stochastic Gradient Descent :
",acc_sgd )
from sklearn.ensemble import
GradientBoostingClassifier
gbk = GradientBoostingClassifier()
gbk.fit(x_train, y_train)
y_pred = gbk.predict(x_val)
acc_gbk =
round(accuracy_score(y_pred, y_val) *
100, 2)
print( "MODEL-10: Accuracy of
GradientBoostingClassifier :
",acc_gbk )
models = pd.DataFrame({
   'Model': ['Logistic
Regression','Gaussian Naive
Bayes','Support Vector Machines',
            'Linear SVC', 'Perceptron',
'Decision Tree',
            'Random Forest',
'KNN','Stochastic Gradient
Descent',
            'Gradient Boosting
Classifier'],
   'Score': [acc_logreg, acc_gaussian,
acc_svc,
            acc_linear_svc,
acc_perceptron,  acc_decisiontree,
            acc_randomforest,  acc_knn,
acc_sgd, acc_gbk]
                  })
print()
print( models.sort_values(by='Score',
```

```
ascending=False) )
ids = test['PassengerId']
predictions =
randomforest.predict(test.drop('Passen
gerId', axis=1))
output = pd.DataFrame({
'PassengerId' : ids, 'Survived':
predictions })
output.to_csv('submission.csv',
index=False)
print( "All survival predictions
done." )
print( "All predictions exported to
submission.csv file." )
print( output )
```

References

www.kaggle.com 15/06/2019

https://en.wikipedia.org/wiki/Machine_learning 20/06/2019