TITANIC SURVIVAL PREDICTION

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.

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.

Here we try to predict which passengers were likely to survive.

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Data Description

The data has been collected from Kaggle Dataset.
(LINK - https://www.kaggle.com/c/titanic)

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include gender_submission.csv, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Variable Notes

pclass: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way…(no. Of parents/children)

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

PREETI CHORARIA
(preeti.choraria0@gmail.com)

TRAINING DATA

test

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292		Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7		S
894	2	Myles, Mr. Thomas Francis	male	62	0	0	240276	9.6875		Q
895	3	Wirz, Mr. Albert	male	27	0	0	315154	8.6625		S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22	1	1	3101298	12.2875		S
897	3	Svensson, Mr. Johan Cervin	male	14	0	0	7538	9.225		S
898	3	Connolly, Miss. Kate	female	30	0	0	330972	7.6292		Q
899	2	Caldwell, Mr. Albert Francis	male	26	1	1	248738	29		S
900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18	0	0	2657	7.2292		C
901	3	Davies, Mr. John Samuel	male	21	2	0	A/4 48871	24.15		S
902	3	Ilieff, Mr. Ylio	male		0	0	349220	7.8958		S
903	1	Jones, Mr. Charles Cresson	male	46	0	0	694	26		S
904	1	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	female	23	1	0	21228	82.2667	B45	S
905	2	Howard, Mr. Benjamin	male	63	1	0	24065	26		S
906	1	Chaffee, Mrs. Herbert Fuller (Carrie Constance Toogood)	female	47	1	0	W.E.P. 5734	61.175	E31	S
907	2	del Carlo, Mrs. Sebastiano (Argenia Genovesi)	female	24	1	0	SC/PARIS 2167	27.7208		C
908	2	Keane, Mr. Daniel	male	35	0	0	233734	12.35		Q
909	3	Assaf, Mr. Gerios	male	21	0	0	2692	7.225		C
910	3	Ilmakangas, Miss. Ida Livija	female	27	1	0	STON/O2. 3101270	7.925		S
911	3	Assaf Khalil, Mrs. Mariana (Miriam")"	female	45	0	0	2696	7.225		C
912	1	Rothschild, Mr. Martin	male	55	1	0	PC 17603	59.4		C
913	3	Olsen, Master. Artur Karl	male	9	0	1	C 17368	3.1708		S
914	1	Flegenheim, Mrs. Alfred (Antoinette)	female		0	0	PC 17598	31.6833		S
915	1	Williams, Mr. Richard Norris II	male	21	0	1	PC 17597	61.3792		C
916	1	Ryerson, Mrs. Arthur Larned (Emily Maria Borie)	female	48	1	3	PC 17608	262.375	B57 B59 B63 B66	C
917	3	Robins, Mr. Alexander A	male	50	1	0	A/5. 3337	14.5		S
918	1	Ostby, Miss. Helene Ragnhild	female	22	0	1	113509	61.9792	B36	C
919	3	Daher, Mr. Shedid	male	22.5	0	0	2698	7.225		C

PREETI CHORARIA
(preeti.choraria0@gmail.com)

TEST DATA

train

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292		Q
24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5	A6	S
25	0	3	Palsson, Miss. Torborg Danira	female	8	3	1	349909	21.075		S
26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)	female	38	1	5	347077	31.3875		S
27	0	3	Emir, Mr. Farred Chehab	male		0	0	2631	7.225		C
28	0	1	Fortune, Mr. Charles Alexander	male	19	3	2	19950	263	C23 C25 C27	S
29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female		0	0	330959	7.8792		Q

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Cleaning:

Removed NAN data,
Categorical to numerical values

Gender	**Numerical Value**
Male	0
Female	1

Embarked	**Numerical Value**
S= Southampton	0
C = Cherbourg	1
Q = Queenstown	2
Others	3

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Correlation

Using Spearman

	Pclass	Sex_cleaned	Age	SibSp	Parch
Pclass	1.000000	-0.159372	-0.361666	-0.050959	-0.018489
Sex_cleaned	-0.159372	1.000000	-0.083330	0.164887	0.254697
Age	-0.361666	-0.083330	1.000000	-0.182061	-0.254212
SibSp	-0.050959	0.164887	-0.182061	1.000000	0.426955
Parch	-0.018489	0.254697	-0.254212	0.426955	1.000000
Fare	-0.730578	0.274996	0.135051	0.422994	0.407150
Embarked_cleaned	-0.163433	0.114816	0.014934	0.011223	0.011576

	Fare	Embarked_cleaned
Pclass	-0.730578	-0.163433
Sex_cleaned	0.274996	0.114816
Age	0.135051	0.014934
SibSp	0.422994	0.011223
Parch	0.407150	0.011576
Fare	1.000000	0.169879
Embarked_cleaned	0.169879	1.000000

Using Pearson's Method of Correlation

	Pclass	Sex_cleaned	Age	SibSp	Parch
Pclass	1.000000	-0.155460	-0.369226	0.067247	0.025683
Sex_cleaned	-0.155460	1.000000	-0.093254	0.103950	0.246972
Age	-0.369226	-0.093254	1.000000	-0.308247	-0.189119
SibSp	0.067247	0.103950	-0.308247	1.000000	0.383820
Parch	0.025683	0.246972	-0.189119	0.383820	1.000000
Fare	-0.554182	0.184994	0.096067	0.138329	0.205119
Embarked_cleaned	-0.125180	0.112058	0.031556	-0.003963	-0.020747

	Fare	Embarked_cleaned
Pclass	-0.554182	-0.125180
Sex_cleaned	0.184994	0.112058
Age	0.096067	0.031556
SibSp	0.138329	-0.003963
Parch	0.205119	-0.020747
Fare	1.000000	0.182383
Embarked_cleaned	0.182383	1.000000

Dimensionality reduction

Removed unwanted columns using feature selection method i.e

1. Name
2. Ticket type
   Were removed.

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Final Cleaned data

PassengerId	Survived	Pclass	Sex	Age	Sibsp	Parch	Fare	Embarked
1	0	3	0	22	1	0	7.25	0
2	1	1	1	38	1	0	71.2833	1
3	1	3	1	26	0	0	7.925	0
4	1	1	1	35	1	0	53.1	0
5	0	3	0	35	0	0	8.05	0
7	0	1	0	54	0	0	51.8625	0
8	0	3	0	2	3	1	21.075	0
9	1	3	1	27	0	2	11.1333	0
10	1	2	1	14	1	0	30.0708	1
11	1	3	1	4	1	1	16.7	0
12	1	1	1	58	0	0	26.55	0
13	0	3	0	20	0	0	8.05	0
14	0	3	0	39	1	5	31.275	0
15	0	3	1	14	0	0	7.8542	0
16	1	2	1	55	0	0	16	0
17	0	3	0	2	4	1	29.125	2
19	0	3	1	31	1	0	18	0
21	0	2	0	35	0	0	26	0
22	1	2	0	34	0	0	13	0
23	1	3	1	15	0	0	8.0292	2
24	1	1	0	28	0	0	35.5	0
25	0	3	1	8	3	1	21.075	0
26	1	3	1	38	1	5	31.3875	0
28	0	1	0	19	3	2	263	0
31	0	1	0	40	0	0	27.7208	1
34	0	2	0	66	0	0	10.5	0
35	0	1	0	28	1	0	82.1708	1
36	0	1	0	42	1	0	52	0
38	0	3	0	21	0	0	8.05	0

PREETI CHORARIA
(preeti.choraria0@gmail.com)

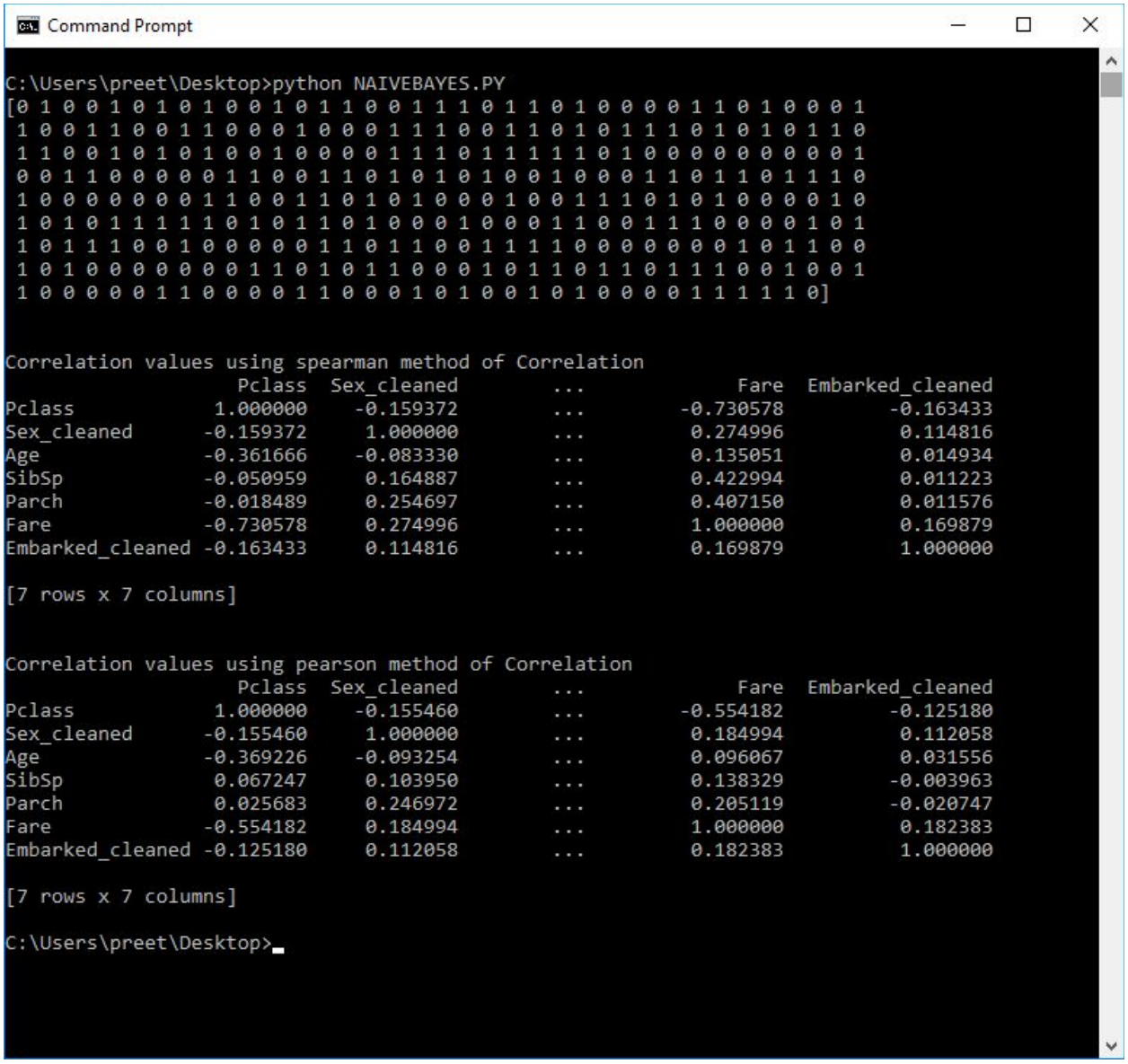
Code for Naive Bayes Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
data = pd.read_csv("train.csv")

#categorical to numerical
data["Sex_cleaned"]=np.where(data["Sex"]=="male",0,1)
data["Embarked_cleaned"]=np.where(data["Embarked"]=="S",0,
                                  np.where(data["Embarked"]=="C",1,
                                           np.where(data["Embarked"]=="Q",2,3)
                                          )
                                 )

# remove NAN values

data=data[[
    "Survived",
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]].dropna(axis=0, how='any')
```

```
gnb = GaussianNB()
used_features =[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]

##   X_train, X_test = train_test_split(data, test_size=0.5, random_state=int(time.time()))

X_train = data

X_test = pd.read_csv("test.csv")



#categorical to numerical
X_test["Sex_cleaned"]=np.where(X_test["Sex"]=="male",0,1)
X_test["Embarked_cleaned"]=np.where(X_test["Embarked"]=="S",0,
                                    np.where(X_test["Embarked"]=="C",1,
                                             np.where(X_test["Embarked"]=="Q",2,3)
                                            )
                                   )

# remove NAN values

X_test=X_test[[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
```

```
    "Parch",
    "Fare",
    "Embarked_cleaned"
]].dropna(axis=0, how='any')
gnb = GaussianNB()
used_features =[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]
# Train classifier
gnb.fit(
    X_train[used_features].values,
    X_train["Survived"]
)
y_pred = gnb.predict(X_test[used_features])
print(y_pred)
```

Result for Naive Bayes Algorithm



Code for Random Forest Algorithm

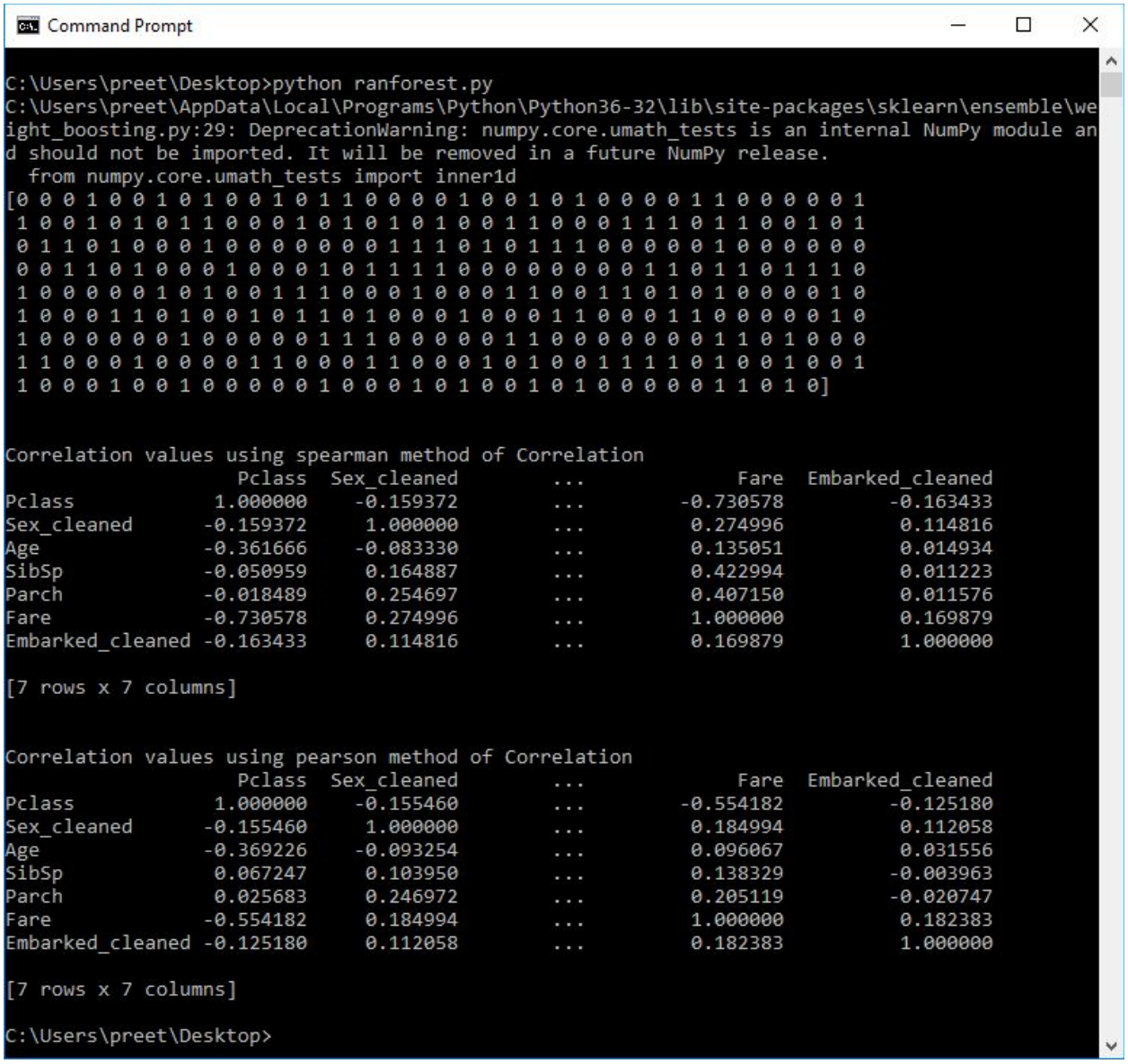
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
data = pd.read_csv("train.csv")

#categorical to numerical
data["Sex_cleaned"]=np.where(data["Sex"]=="male",0,1)
data["Embarked_cleaned"]=np.where(data["Embarked"]=="S",0,
                                  np.where(data["Embarked"]=="C",1,
                                           np.where(data["Embarked"]=="Q",2,3)
                                          )
                                 )

# remove NAN values

data=data[[
    "Survived",
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
```

```
    "Parch",
    "Fare",
    "Embarked_cleaned"
]].dropna(axis=0, how='any')
used_features =[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]

##   X_train, X_test = train_test_split(data, test_size=0.5, random_state=int(time.time()))

X_train = data

X_test = pd.read_csv("test.csv")



#categorical to numerical
X_test["Sex_cleaned"]=np.where(X_test["Sex"]=="male",0,1)
X_test["Embarked_cleaned"]=np.where(X_test["Embarked"]=="S",0,
                                    np.where(X_test["Embarked"]=="C",1,
                                             np.where(X_test["Embarked"]=="Q",2,3)
                                            )
                                   )

# remove NAN values

X_test=X_test[[
    "Pclass",
```

```
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]].dropna(axis=0, how='any')

used_features =[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]

def random_forest_classifier(features, target):
    """
    To train the random forest classifier with features and target data
    :param features:
    :param target:
    :return: trained random forest classifier
    """
    clf = RandomForestClassifier()
    clf.fit(features, target)
    return clf

trained_model = random_forest_classifier(X_train[used_features], X_train["Survived"])
predictions = trained_model.predict(X_test[used_features])
print(predictions)
```

PREETI CHORARIA
(preeti.choraria0@gmail.com)

Result for Random Forest Algorithm



Code for RNN

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
feature_sets_train = pd.read_csv('train.csv')
# TODO: Use both datasets to make the embeddings (vocab_to_int map)
feature_sets_test = pd.read_csv('test.csv')
feature_sets_train_tests = pd.concat([feature_sets_train, feature_sets_test])
feature_sets = feature_sets_train
passengers = [' '.join(map(str,passenger[[2,3,4,5,8,9,10,11]])) for passenger in
feature_sets.values]
passengers_test = [' '.join(map(str,passenger[[1,2,3,4,7,8,9,10]])) for passenger in
feature_sets_test.values]


survived = [passenger[1] for passenger in feature_sets.values]
feature_sets = passengers
feature_sets_test = passengers_test
labels =  survived
###feature_sets_train
###feature_sets_train_tests
#from string import punctuation
#all_text = ''.join([c for c in feature_sets if c not in punctuation])
#feature_sets = all_text.split(',')


passengers = [' '.join(map(str,passenger[[0,1,2,3,4,5,7,8,9,11]])) for passenger in
feature_sets_train_tests.values]


all_text = ' '.join(passengers)
```

```
words = all_text.split()
###all_text[:1000]
###words[:10]
from collections import Counter
counts = Counter(words)
vocab = sorted(counts, key=counts.get, reverse=True)
vocab_to_int = {word: ii for ii, word in enumerate(vocab, 1)}

feature_sets_ints = []
feature_sets_ints_test = []
print(feature_sets[0])
for each in feature_sets:
    feature_sets_ints.append([vocab_to_int[word] for word in each.split()])

print(feature_sets_test[0])
for each in feature_sets_test:
    feature_sets_ints_test.append([vocab_to_int[word] for word in each.split()])
feature_set_lens = Counter([len(x) for x in feature_sets_ints])
print("Zero-length feature_sets: {}".format(feature_set_lens[0]))
print("Maximum feature_set length: {}".format(max(feature_set_lens)))
non_zero_idx = [ii for ii, feature_set in enumerate(feature_sets_ints) if len(feature_set) != 0]
print(len(non_zero_idx))

non_zero_idx_test = [ii for ii, feature_set in enumerate(feature_sets_ints_test) if len(feature_set)
!= 0]
print(len(non_zero_idx_test))
print(feature_sets_ints[-1])
print(feature_sets_ints_test[-1])

feature_sets_ints = [feature_sets_ints[ii] for ii in non_zero_idx]
feature_sets_ints_test = [feature_sets_ints_test[ii] for ii in non_zero_idx_test]

labels = np.array([labels[ii] for ii in non_zero_idx])
```

```
seq_len = 24
features = np.zeros((len(feature_sets_ints), seq_len), dtype=int)
for i, row in enumerate(feature_sets_ints):
    features[i, -len(row):] = np.array(row)[:seq_len]
seq_len = 24
features_test = np.zeros((len(feature_sets_ints_test), seq_len), dtype=int)
for i, row in enumerate(feature_sets_ints_test):
    features_test[i, -len(row):] = np.array(row)[:seq_len]

split_frac = 0.75
split_idx = int(len(features)*split_frac)
train_x, val_x = features[:split_idx], features[split_idx:]
train_y, val_y = labels[:split_idx], labels[split_idx:]

test_idx = int(len(val_x)*0.5)
val_x, test_x = val_x[:test_idx], val_x[test_idx:]
val_y, test_y = val_y[:test_idx], val_y[test_idx:]

print("\t\t\tFeature Shapes:")
print("Train set: \t\t{}".format(train_x.shape),
      "\nValidation set: \t{}".format(val_x.shape),
      "\nTest set: \t\t{}".format(test_x.shape))

lstm_size = 256
lstm_layers = 1
batch_len = 100
learning_rate = 0.001

n_words = len(vocab_to_int)+1

# Create the graph object
```

```
graph = tf.Graph()
# Add nodes to the graph
with graph.as_default():
    inputs_ = tf.placeholder(tf.int32, [None, None], name='inputs')
    labels_ = tf.placeholder(tf.int32, [None, None], name='labels')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')
    batch_size = tf.placeholder_with_default(tf.constant(batch_len), shape=[], name='batch_size')
```