Data Mining & Analytics: Kaggle Competition

Group: Ashley Santos, Warren Lim, Ian Castro, Arushi Sharma

```
import pandas as pd
import numpy as np

# Model libraries
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# Preprocessing tools
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import normalize
from sklearn.feature_extraction import DictVectorizer


# Misc. sklearn tools
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, ParameterGrid
from sklearn.model_selection import KFold

import warnings
warnings.filterwarnings("ignore")


# Ensemble
from sklearn.ensemble import VotingClassifier
```

```
# Data viz
import seaborn as sns
import matplotlib.pyplot as plt
```

Data pre-processing steps

```
from google.colab import drive
drive.mount("/content/drive/")
```

```
    Mounted at /content/drive/
```

```
%cd /content/drive/My Drive/Colab Notebooks/
```

```
    /content/drive/My Drive/Colab Notebooks
```

```
train = pd.read_csv('data/kaggle/train.csv')
test = pd.read_csv('data/kaggle/test.csv')
```

```
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

```
test.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

What is the majority class of `Survived`?

```
train['Survived'].value_counts()
```

```
0    549
1    342
Name: Survived, dtype: int64
```

EDA

```
# People in first class tend to be older than people in third class
train.groupby("Pclass").agg(np.mean)
```

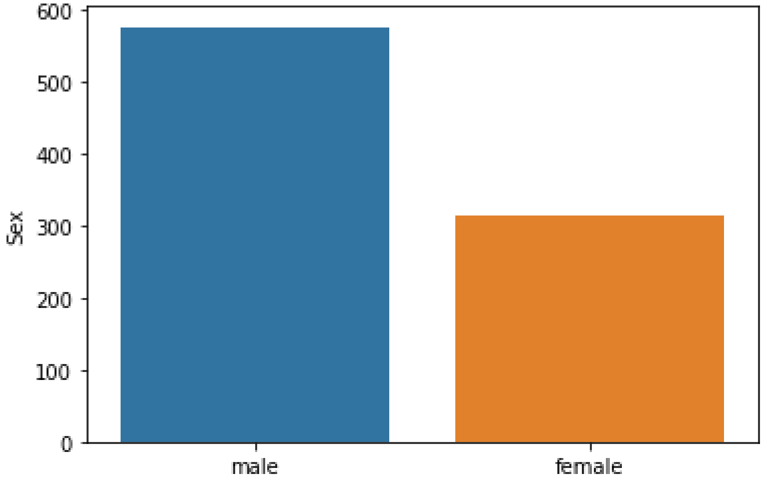
	PassengerId	Survived	Age	SibSp	Parch	Fare
Pclass						
1	461.597222	0.629630	38.233441	0.416667	0.356481	84.154687
2	445.956522	0.472826	29.877630	0.402174	0.380435	20.662183
3	439.154786	0.242363	25.140620	0.615071	0.393075	13.675550

```
train.groupby("Survived").agg(np.mean)
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
Survived						
0	447.016393	2.531876	30.626179	0.553734	0.329690	22.117887
1	444.368421	1.950292	28.343690	0.473684	0.464912	48.395408

```
# vars to check: sex, cabin
print(train["Sex"].value_counts(), "\n")

sex_data = train["Sex"].value_counts()
sns.barplot(x = sex_data.index, y = sex_data);
```

```
male      577
female    314
Name: Sex, dtype: int64
```



```
pd.crosstab(train["Sex"], train["Survived"])
```

Survived	0	1
Sex		
female	81	233

```
print("Cabin has this many nulls:", sum(train["Cabin"].isna()), "and this many rows: ", len(train["Cabin"]))

# NOTE: this many nulls means it's probably not a great variable to use?
```

```
    Cabin has this many nulls: 687 and this many rows:  891
```

```
# We want only the letters that designate the location
known_cabins = train["Cabin"].str[:1]
train["Cabin Letter"] = train["Cabin"].str[:1]

known_cabins.value_counts()
```

```
    C    59
    B    47
    D    33
    E    32
    A    15
    F    13
    G     4
    T     1
    Name: Cabin, dtype: int64
```

```
pd.crosstab(train["Cabin Letter"], train["Survived"])
```

Survived	0	1
Cabin Letter		
A	8	7
B	12	35
C	24	35

```
train["Pclass"].value_counts()

pd.crosstab(train["Cabin Letter"], train["Pclass"])

# cabin letter = related to class
# wealthiest people survived AND lived in cabins A-E
# @Ian: create variable that says 1 = in A-E, 0 = NA or anywhere else
```

Pclass	1	2	3
Cabin Letter			
A	15	0	0
B	47	0	0
C	59	0	0
D	29	4	0
E	25	4	3
F	0	8	5
G	0	0	4
T	1	0	0

```
pd.crosstab(train["Pclass"], train["Survived"])
```

Survived	0	1
Pclass		
1	80	136
2	97	87

```
train["Cabin Letter"].isin(["A", "B", "C", "D", "E"])
```

```
0      False
1       True
2      False
3       True
4      False
       ...  
886    False
887     True
888    False
889     True
890    False
Name: Cabin Letter, Length: 891, dtype: bool
```

```
train["Cabin Letter"] = train["Cabin"].str[:1]
train["top_decks"] = train["Cabin Letter"].isin(["A", "B", "C", "D", "E"])
```

```
# siblings EDA: number of siblings on the titanic

train["SibSp"][train["SibSp"].isna()] # No Nulls
```

```
Series([], Name: SibSp, dtype: int64)
```

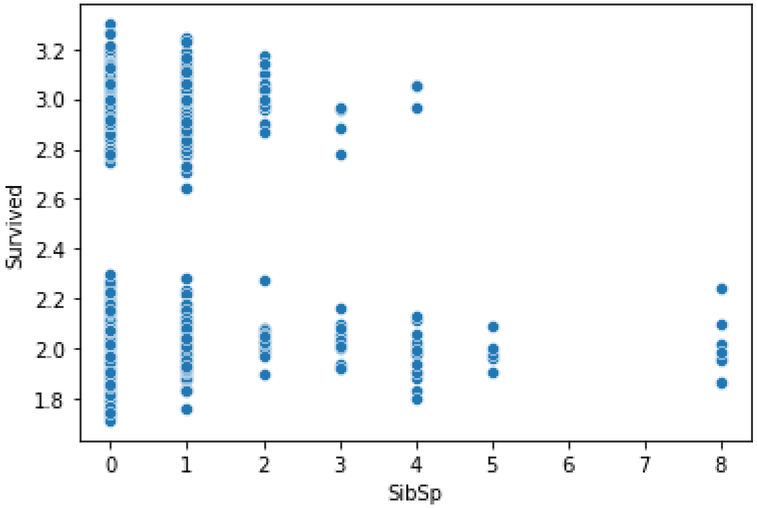
```
print(np.mean(train["SibSp"]))

train["SibSp"].value_counts()
```

```
0.5230078563411896
0    608
```

```
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
sns.scatterplot(x = train["SibSp"], y = train["Survived"] + np.random.normal(2, 0.1, train["Survived"].shape))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa74db7b190>
```



```
pd.crosstab(train["SibSp"], train["Survived"])
```

Survived	0	1
SibSp		
0	398	210

Feature Engineering

3	12	4

```
# Individuals that are in the top decks tend to survive
decks = ["A", "B", "C", "D", "E"]
top_decks = train["Cabin"].str[:1].isin(decks)
train["top_decks"] = top_decks
train["Cabin_letter"] = train["Cabin"].str[:1]


# Making PClass a categorical variable for dummy analysis
train["Pclass_cat"] = train["Pclass"].replace({1: "First", 2: "Second", 3: "Third"})


# Imputing to class average; ignore for this version
#train.loc[(train["Age"].isnull()) & (train["Pclass"] == 1), "Age"] = np.mean(train.loc[(~train["Age"].isnull()) &
#train.loc[(train["Age"].isnull()) & (train["Pclass"] == 2), "Age"] = np.mean(train.loc[(~train["Age"].isnull()) &
#train.loc[(train["Age"].isnull()) & (train["Pclass"] == 3), "Age"] = np.mean(train.loc[(~train["Age"].isnull()) &


train["Alone"] = (train["Parch"] == 0)


def z_score(data):
  return (data - np.mean(data)) / np.std(data)

train["Fare_z"] = z_score(train["Fare"])
train["Age_z"] = z_score(train["Age"])
train["Parch_z"] = z_score(train["Parch"])
train["SibSp_z"] = z_score(train["SibSp"])
```

Our approach

```
features = ["Survived", "Pclass", "Age", "Sex"]

# Using this to pull vars of interest from train and test sets
X_train = pd.get_dummies(train[features].fillna(train.mean()))

Y_train = X_train["Survived"]

X_train_features = X_train.drop("Survived", axis = 1)


hidden_layers = (10)
clf = MLPClassifier(hidden_layer_sizes=hidden_layers, activation = 'logistic',
                    solver='lbfgs', random_state=42)

# Training/testing
clf.fit(X_train_features,Y_train)

# Making predictions
y_preds = clf.predict(X_train_features)

# Evaluating accuracy
sum(y_preds == Y_train) / len(Y_train)
```

```
    0.8013468013468014
```

```
X_test = test.fillna(test.mean())
X_test_features = pd.get_dummies(X_test[features[1:]])

y_test_preds = clf.predict(X_test_features)

final_predictions = X_test[["PassengerId"]]
final_predictions["Survived"] = y_test_preds

final_predictions.to_csv('my_prediction.csv')
```

Cross Validation: Evaluating the Model

```
kf = KFold(n_splits = 5)

cv_scores = []

for train_indices, val_indices in kf.split(X_train):
  X_train_cv = X_train_features.loc[train_indices] # no labels, just features for training set portion
  Y_train_cv = Y_train[train_indices] # labels for training set portion
  #clf.fit(X_train_cv, Y_train_cv) # train on subset of training set
  X_val_cv = X_train_features.loc[val_indices]
  Y_val_cv = Y_train[val_indices]
  cv_scores.append(clf.score(X_val_cv, Y_val_cv)) # test on validation set and make a score

print(np.mean(cv_scores))
cv_scores
```

```
0.8013746783001693
[0.776536312849162,
 0.8033707865168539,
 0.7921348314606742,
 0.7752808988764045,
 0.8595505617977528]
```

```
print("This is the average cross-validated score for this model:", np.mean(cv_scores))
```

```
This is the average cross-validated score for this model: 0.8013746783001693
```

To create a submission for the Kaggle Competition, use `to_csv` with the name of your file output. An example is below. Your file will be output in your current directory. On Google Colab, by default, this is `content`.

Download this csv and upload it to Kaggle. Good luck! We are rooting for you :)

✓ 0s completed at 9:15 PM