Lab 5: Kaggle Competition

Notebook by: Davis Ulrich

Partner: Bridget Silver

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import silhouette_score
from sklearn.model_selection import KFold

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_extraction import DictVectorizer

from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
```

```
train = pd.read_csv('/content/drive/My Drive/Data 144/train.csv')
test = pd.read_csv("/content/drive/My Drive/Data 144/test.csv")
train
```

	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
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
				Allen, Mr								

```
test
```

Saved successfully! ×

			Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
			elly, 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
...	...	...	...	...	...	...	...	...	...	...	...

Dealing with Nans

Dona. Fermina

```
train.isna().sum()
```

```
    PassengerId      0
    Survived         0
    Pclass           0
    Name             0
    Sex              0
    Age            177
    SibSp            0
    Parch            0
    Ticket           0
    Fare             0
    Cabin          687
    Embarked         2
    dtype: int64
```

```
test.isna().sum()
```

```
    PassengerId      0
    Pclass           0
    Name             0
    Sex              0
    Age             86
    SibSp            0
    Parch            0
    Ticket           0
    Fare             1
    Cabin          327
    Embarked         0
    dtype: int64
```

```
# The Fare feature is a weird object that you can't average or anyting
train['Fare'] = [float(i) for i in train['Fare']]
test['Fare'] = [float(i) for i in test['Fare']]
```

```
# Many more people who have a missing age did not survive
train.sort_values('Age').iloc[800:]['Survived'].mean()
```

```
    0.21978021978021978
```

```
# Since most passengers missing their age didn't survive,
# it's relevant to keep those columns and be able to distinguish them
train['Age'] = train[['Age']].fillna(999.0)
```

```
test['Age'] = test[['Age']].fillna(999.0)
```

Saved successfully!

```
lna(test['Fare'].mean())

train = train.fillna('NA')
test = test.fillna('NA')

# train = train.dropna()
# test = test.dropna()
train
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NA	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NA	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
				Allen, Mr								

Adding Features / One-Hot Encoding

```
train['Sex'] = [1 if sex == 'male' else 0 for sex in train['Sex']]
test['Sex'] = [1 if sex == 'male' else 0 for sex in test['Sex']]
train
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NA	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	NA	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	C123	S
				Allen, Mr								

```
# Out of those that have long names (greater than 35 characters), 71% survived.
                                          ow in train['Name']]]['Survived'].mean())
```

Saved successfully! ×

```
# train['long_name'] = [int(len(name) >= 35) for name in train['Name']]
# test['long_name'] = [int(len(name) >= 35) for name in test['Name']]
train
```

```
0.7071428571428572
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NA	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	NA	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	C123	S
				Allen, Mr								

```
# Length of name includes long and short
train['name_len'] = [len(name) for name in train['Name']]
test['name_len'] = [len(name) for name in test['Name']]


# Some embarking locations have higher average survival rate than others
print(train[train['Embarked'] == 'S']['Survived'].mean())

train = train.merge(pd.get_dummies(train['Embarked']), left_index = True, right_index = True)
test = test.merge(pd.get_dummies(test['Embarked']), left_index = True, right_index = True)
train
```

```
0.33695652173913043
```

Saved successfully!

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

```
data = [train, test]
titles = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Not Common": 5}

for dataset in data:
    # extract titles
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr','Major', 'Rev'

    # convert titles into numbers
    dataset['Title'] = dataset['Title'].map(titles)

    dataset['Title'] = dataset['Title'].fillna(0)
train = train.drop(['Name'], axis=1)
test = test.drop(['Name'], axis=1)
```

```
data = [train, test]
for dataset in data:
    dataset['number relatives'] = dataset['SibSp'] + dataset['Parch']
```

```
data = [train, test]
for dataset in data:
    dataset.loc[dataset['number relatives'] > 0, 'Alone?'] = 0
    dataset.loc[dataset['number relatives'] == 0, 'Alone?'] = 1
    dataset['Alone?'] = dataset['Alone?'].astype(int)
```

Train Validation Split

```
# Train Val Split
X_train, X_val, Y_train, Y_val = train_test_split(train.drop(['Survived'], axis = 1), train['Survived'], ran
```

Saved successfully! ×

Models

Model 1: Decision Tree

```
train.columns
```

```
     Index(['PassengerId', 'Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch',
            'Ticket', 'Fare', 'Cabin', 'Embarked', 'name_len', 'C', 'NA', 'Q', 'S',
            'Title', 'number relatives', 'Alone?'],
           dtype='object')
```

```
all_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'name_len',
       'Title', 'number relatives', 'Alone?']
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'name_len', 'C', 'Q', 'S', 'NA', 'Title', 'num

tree = DecisionTreeClassifier(random_state = 42)
tree.fit(X_train[features], Y_train)

y_pred_tree = tree.predict(X_val[features])
accuracy_score(Y_val, y_pred_tree)
```

```
     0.7802690582959642
```

```
# Best so far

# 73.5 ['Pclass', 'Sex', 'Age', 'Fare']
# 78.5 ['Sex']
# 80.7 ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'long_name', 'name_len', 'C', 'Q', 'S']
```

```
all_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'name_len', 'C', 'Q', 'S', 'NA']
featuresm = ['Sex', 'Age', 'Fare']

tree2 = DecisionTreeClassifier(max_depth=1, random_state = 42)
tree2.fit(X_train[featuresm], Y_train)

y_pred_tree2 = tree2.predict(X_val[featuresm])
accuracy_score(Y_val, y_pred_tree2)
```

```
     0.7847533632286996
```

Model 2: Neural Network

Saved successfully!

```
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
                'name_len', 'C', 'Q', 'S']

NN = MLPClassifier(hidden_layer_sizes = (10), activation = 'logistic', solver = 'lbfgs', max_iter=300, rando
NN.fit(X_train[features], Y_train)

y_pred_NN = NN.predict(X_val[features])
accuracy_score(Y_val, y_pred_NN)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:470: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
0.7847533632286996
```

Model 3: Ensemble Learning

```
# Taking the majority vote from the two decision trees and the neural network

ensemble = y_pred_tree + y_pred_tree2 + y_pred_NN
ensemble = np.round(ensemble/3, 0)
ensemble
```

```
array([1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1.,
       0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1.,
       1., 0., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0.,
       0., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 0.,
       0., 1., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 1.,
       1., 0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1.,
       1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 1., 1., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0.,
       1., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0.,
       1., 0.])
```

```
accuracy_score(Y_val, ensemble)
```

```
0.8071748878923767
```

Saved successfully! ×

Testing

```
test
```

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	name_len	C	Q
0	892	3	1	34.5	0	0	330911	7.8292	NA	Q	16	0	1
1	893	3	0	47.0	1	0	363272	7.0000	NA	S	32	0	0
2	894	2	1	62.0	0	0	240276	9.6875	NA	Q	25	0	1
3	895	3	1	27.0	0	0	315154	8.6625	NA	S	16	0	0
4	896	3	0	22.0	1	1	3101298	12.2875	NA	S	44	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	1	999.0	0	0	A.5. 3236	8.0500	NA	S	18	0	0
414	1306	1	0	39.0	0	0	PC 17758	108.9000	C105	C	28	1	0
415	1307	3	1	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NA	S	28	0	0
416	1308	3	1	999.0	0	0	359309	8.0500	NA	S	19	0	0
417	1309	3	1	999.0	1	1	2668	22.3583	NA	C	24	1	0

```
test[features]
```

	Pclass	Sex	Age	SibSp	Parch	Fare	name_len	C	Q	S
0	3	1	34.5	0	0	7.8292	16	0	1	0
1	3	0	47.0	1	0	7.0000	32	0	0	1
2	2	1	62.0	0	0	9.6875	25	0	1	0
3	3	1	27.0	0	0	8.6625	16	0	0	1
4	3	0	22.0	1	1	12.2875	44	0	0	1
...	...	...	...	...	...	...	...	...	...	...
413	3	1	999.0	0	0	8.0500	18	0	0	1
414	1	0	39.0	0	0	108.9000	28	1	0	0
415	3	1	38.5	0	0	7.2500	28	0	0	1
416	3	1	999.0	0	0	8.0500	19	0	0	1
417	3	1	999.0	1	1	22.3583	24	1	0	0

418 rows × 10 columns

```
NN.predict(test[features])
```

```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
                              0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
                              0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
                              1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0])
```

Saved successfully! ×

```
submission = pd.DataFrame(index=test.PassengerId)
# submission['Survived'] = NN.predict(test[features])
submission['Survived'] = tree2.predict(test[featuresm])
submission
```

```
                Survived
 PassengerId
     892               0
     893               1
     894               0
     895               0
     896               1
     ...             ...
    1305               0
    1306               1
    1307               0
    1308               0
    1309               0

418 rows × 1 columns
```

```
submission.reset_index().to_csv('submission.csv', index=False)
```

Saved successfully! ×