In [ ]:
```
# General data analysis/plotting
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Data preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Neural Net modules
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [ ]:
```
df = pd.read_csv("https://github.com/YBI-Foundation/Dataset/raw/main/Titanic.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   pclass     1309 non-null   int64  
 1   survived   1309 non-null   int64  
 2   name       1309 non-null   object 
 3   sex        1309 non-null   object 
 4   age        1046 non-null   float64
 5   sibsp      1309 non-null   int64  
 6   parch      1309 non-null   int64  
 7   ticket     1309 non-null   object 
 8   fare       1308 non-null   float64
 9   cabin      295 non-null    object 
 10  embarked   1307 non-null   object 
 11  boat       486 non-null    object 
 12  body       121 non-null    float64
 13  home.dest  745 non-null    object 
dtypes: float64(3), int64(4), object(7)
memory usage: 143.3+ KB
```

In [ ]:
```
df.drop(['body','home.dest','boat','cabin'],axis=1,inplace=True)
df.isna().sum()
```

Out[ ]:
```
pclass        0
survived      0
name          0
sex           0
age         263
sibsp         0
parch         0
ticket        0
fare          1
embarked      2
dtype: int64
```

In [ ]:
```
df['age'].fillna(round(df['age'].mean()),inplace=True)
df['fare'].fillna(df['fare'].median(),inplace=True)
df['embarked'].fillna(df['embarked'].mode()[0],inplace=True)
```

In [ ]:
```
df.replace({'sex':{'male':0,'female':1}, 'embarked':{'S':0,'C':1,'Q':2}}, inplace=T
```

In [ ]:
```
X = df.drop(columns = ['name','ticket','survived'],axis=1)
y=df['survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
```

In [ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

svc = SVC(C=1.0, random_state=1, kernel='linear')
svc.fit(X_train, y_train)
pred = svc.predict(X_test)
print('Support Vector Classifier Accuracy', accuracy_score(pred, y_test))
```

```
Support Vector Classifier Accuracy 0.7557251908396947
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=42)
dt.fit(X_train, y_train)
pred = dt.predict(X_test)
print('Decision Tree Classifier Accuracy', accuracy_score(pred, y_test))
```

```
Decision Tree Classifier Accuracy 0.7595419847328244
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 36)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print('K Nearest Neighbors Classifier Accuracy', accuracy_score(pred, y_test))
```

```
K Nearest Neighbors Classifier Accuracy 0.6679389312977099
c:\Users\chinn\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\ba
se.py:441: UserWarning: X does not have valid feature names, but KNeighborsClassif
ier was fitted with feature names
  warnings.warn(
```

In [ ]:
```
model = Sequential()
model.add(Dense(64, input_shape=(X_train.shape[1],), activation='relu')) # (feature
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid')) # output node

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, batch_size
```

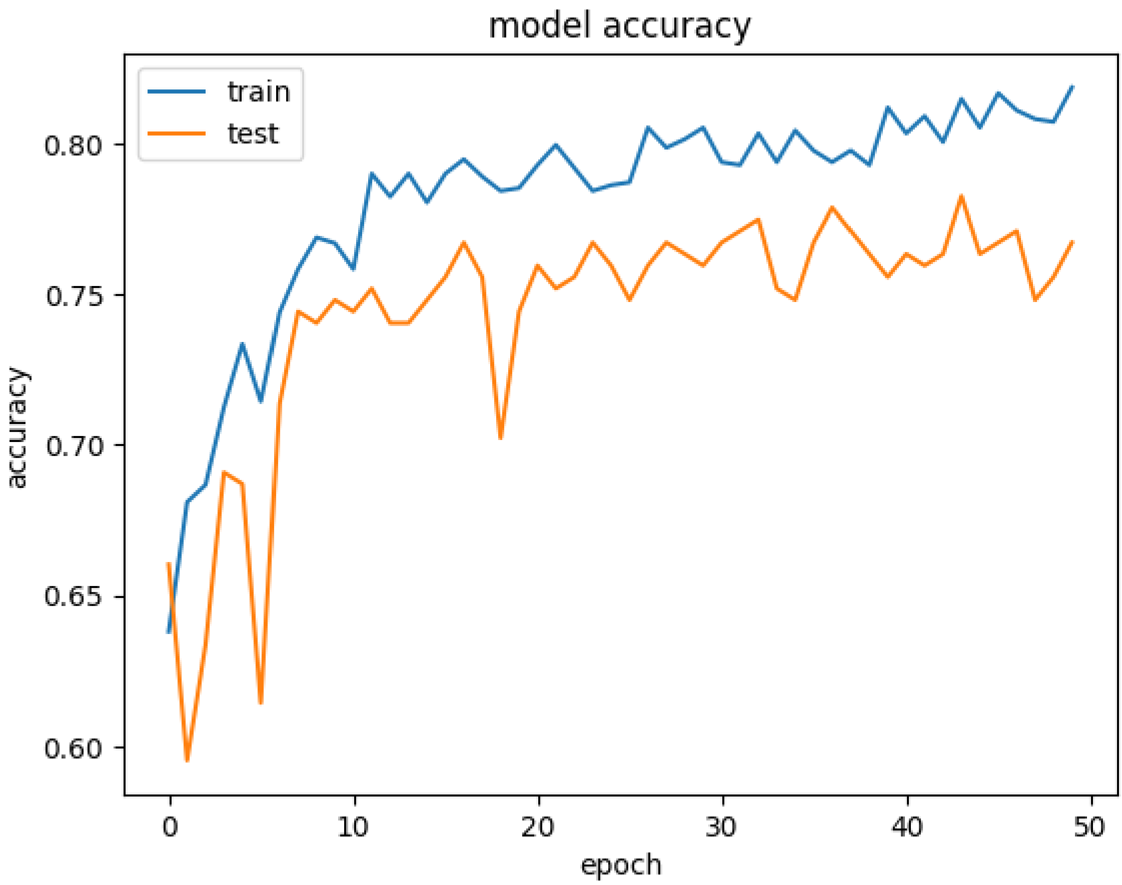
```
Epoch 1/50
66/66 [==============================] - 1s 5ms/step - loss: 0.7536 - accuracy: 0.
6380 - val_loss: 0.6479 - val_accuracy: 0.6603
Epoch 2/50
66/66 [==============================] - 0s 3ms/step - loss: 0.6180 - accuracy: 0.
6810 - val_loss: 0.6669 - val_accuracy: 0.5954
Epoch 3/50
66/66 [==============================] - 0s 3ms/step - loss: 0.5970 - accuracy: 0.
6867 - val_loss: 0.6133 - val_accuracy: 0.6336
Epoch 4/50
66/66 [==============================] - 0s 3ms/step - loss: 0.5826 - accuracy: 0.
7125 - val_loss: 0.5940 - val_accuracy: 0.6908
Epoch 5/50
66/66 [==============================] - 0s 2ms/step - loss: 0.5500 - accuracy: 0.
7335 - val_loss: 0.6011 - val_accuracy: 0.6870
Epoch 6/50
66/66 [==============================] - 0s 3ms/step - loss: 0.5432 - accuracy: 0.
7144 - val_loss: 0.6372 - val_accuracy: 0.6145
Epoch 7/50
66/66 [==============================] - 0s 3ms/step - loss: 0.5422 - accuracy: 0.
7440 - val_loss: 0.5553 - val_accuracy: 0.7137
Epoch 8/50
66/66 [==============================] - 0s 2ms/step - loss: 0.5309 - accuracy: 0.
7584 - val_loss: 0.5183 - val_accuracy: 0.7443
Epoch 9/50
66/66 [==============================] - 0s 2ms/step - loss: 0.5146 - accuracy: 0.
7689 - val_loss: 0.5224 - val_accuracy: 0.7405
Epoch 10/50
66/66 [==============================] - 0s 2ms/step - loss: 0.5191 - accuracy: 0.
7670 - val_loss: 0.5336 - val_accuracy: 0.7481
Epoch 11/50
66/66 [==============================] - 0s 2ms/step - loss: 0.5102 - accuracy: 0.
7584 - val_loss: 0.5320 - val_accuracy: 0.7443
Epoch 12/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4802 - accuracy: 0.
7899 - val_loss: 0.5072 - val_accuracy: 0.7519
Epoch 13/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4820 - accuracy: 0.
7822 - val_loss: 0.5318 - val_accuracy: 0.7405
Epoch 14/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4966 - accuracy: 0.
7899 - val_loss: 0.5292 - val_accuracy: 0.7405
Epoch 15/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4724 - accuracy: 0.
7803 - val_loss: 0.5354 - val_accuracy: 0.7481
Epoch 16/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4642 - accuracy: 0.
7899 - val_loss: 0.4933 - val_accuracy: 0.7557
Epoch 17/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4586 - accuracy: 0.
7947 - val_loss: 0.5054 - val_accuracy: 0.7672
Epoch 18/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4750 - accuracy: 0.
7889 - val_loss: 0.5192 - val_accuracy: 0.7557
Epoch 19/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4747 - accuracy: 0.
7841 - val_loss: 0.6218 - val_accuracy: 0.7023
Epoch 20/50
66/66 [==============================] - 0s 3ms/step - loss: 0.5098 - accuracy: 0.
7851 - val_loss: 0.5713 - val_accuracy: 0.7443
Epoch 21/50
```

```
66/66 [==============================] - 0s 3ms/step - loss: 0.4628 - accuracy: 0.
7927 - val_loss: 0.5016 - val_accuracy: 0.7595
Epoch 22/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4564 - accuracy: 0.
7994 - val_loss: 0.4947 - val_accuracy: 0.7519
Epoch 23/50
66/66 [==============================] - 0s 4ms/step - loss: 0.4592 - accuracy: 0.
7918 - val_loss: 0.5452 - val_accuracy: 0.7557
Epoch 24/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4593 - accuracy: 0.
7841 - val_loss: 0.4978 - val_accuracy: 0.7672
Epoch 25/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4727 - accuracy: 0.
7861 - val_loss: 0.5193 - val_accuracy: 0.7595
Epoch 26/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4750 - accuracy: 0.
7870 - val_loss: 0.5490 - val_accuracy: 0.7481
Epoch 27/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4577 - accuracy: 0.
8052 - val_loss: 0.5040 - val_accuracy: 0.7595
Epoch 28/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4495 - accuracy: 0.
7985 - val_loss: 0.5016 - val_accuracy: 0.7672
Epoch 29/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4504 - accuracy: 0.
8013 - val_loss: 0.4893 - val_accuracy: 0.7634
Epoch 30/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4429 - accuracy: 0.
8052 - val_loss: 0.5415 - val_accuracy: 0.7595
Epoch 31/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4711 - accuracy: 0.
7937 - val_loss: 0.5181 - val_accuracy: 0.7672
Epoch 32/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4512 - accuracy: 0.
7927 - val_loss: 0.4894 - val_accuracy: 0.7710
Epoch 33/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4510 - accuracy: 0.
8032 - val_loss: 0.4977 - val_accuracy: 0.7748
Epoch 34/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4552 - accuracy: 0.
7937 - val_loss: 0.5470 - val_accuracy: 0.7519
Epoch 35/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4392 - accuracy: 0.
8042 - val_loss: 0.5123 - val_accuracy: 0.7481
Epoch 36/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4532 - accuracy: 0.
7975 - val_loss: 0.4890 - val_accuracy: 0.7672
Epoch 37/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4436 - accuracy: 0.
7937 - val_loss: 0.4979 - val_accuracy: 0.7786
Epoch 38/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4348 - accuracy: 0.
7975 - val_loss: 0.4944 - val_accuracy: 0.7710
Epoch 39/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4551 - accuracy: 0.
7927 - val_loss: 0.5152 - val_accuracy: 0.7634
Epoch 40/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4466 - accuracy: 0.
8118 - val_loss: 0.5125 - val_accuracy: 0.7557
Epoch 41/50
66/66 [==============================] - 0s 2ms/step - loss: 0.4399 - accuracy: 0.
```

```
8032 - val_loss: 0.5101 - val_accuracy: 0.7634
Epoch 42/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4336 - accuracy: 0.
8090 - val_loss: 0.4956 - val_accuracy: 0.7595
Epoch 43/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4302 - accuracy: 0.
8004 - val_loss: 0.4902 - val_accuracy: 0.7634
Epoch 44/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4372 - accuracy: 0.
8147 - val_loss: 0.4902 - val_accuracy: 0.7824
Epoch 45/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4415 - accuracy: 0.
8052 - val_loss: 0.5017 - val_accuracy: 0.7634
Epoch 46/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4309 - accuracy: 0.
8166 - val_loss: 0.5304 - val_accuracy: 0.7672
Epoch 47/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4248 - accuracy: 0.
8109 - val_loss: 0.5464 - val_accuracy: 0.7710
Epoch 48/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4330 - accuracy: 0.
8080 - val_loss: 0.5753 - val_accuracy: 0.7481
Epoch 49/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4307 - accuracy: 0.
8071 - val_loss: 0.5777 - val_accuracy: 0.7557
Epoch 50/50
66/66 [==============================] - 0s 3ms/step - loss: 0.4310 - accuracy: 0.
8185 - val_loss: 0.5003 - val_accuracy: 0.7672
```

Out[ ]:
```
<keras.callbacks.History at 0x208f17fdb50>
```

In [ ]:
```
# plot the accuracy
plt.plot(model.history.history['accuracy'])
plt.plot(model.history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```


model accuracy
train
test
0.80
0.75
0.70
0.65
0.60
accuracy
0
10
20
30
40
50
epoch

```
In [ ]:
```