Background:

From Alphabet Soup's business team, Beks received a CSV containing more than 34,000 organizations that have received various amounts of funding from Alphabet Soup over the years. Within this dataset are a number of columns that capture metadata about each organization such as the following:

- EIN and NAME—Identification columns
- APPLICATION_TYPE—Alphabet Soup application type
- AFFILIATION—Affiliated sector of industry
- CLASSIFICATION—Government organization classification
- USE_CASE—Use case for funding
- ORGANIZATION—Organization type
- STATUS—Active status
- INCOME_AMT—Income classification
- SPECIAL_CONSIDERATIONS—Special consideration for application
- ASK_AMT—Funding amount requested
- IS_SUCCESSFUL—Was the money used effectively

Assignment Objectives

The goals of this challenge are:

- Import, analyze, clean, and preprocess a "real-world" classification dataset.

- Select, design, and train a binary classification model of your choosing.

- Optimize model training and input data to achieve desired model performance.

Analysis:

1. How many neurons and layers did you select for your neural network model? Why?
2. Were you able to achieve the target model performance? What steps did you take to try and increase model performance?
3. If you were to implement a different model to solve this classification problem, which would you choose? Why?

Import Dependencies and Data.

Model Comparison Choice:

Random forest model will be selected for comparison and to design optimal training parameters for the deep neural network due to it's robustness of obtaining reliable prediction from large datasets such as this one used for the assignment.

In [1]:

```
# Import our dependencies
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
import pandas as pd
import tensorflow as tf
```

```
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:518: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:519: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:520: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorflow/pytho
n/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1type'
as a synonym of type is deprecated; in a future version of numpy, it wi
ll be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/tensorboard/compa
t/tensorflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or '1
```

```
type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:
```
# Import our input dataset
charity_df = pd.read_csv('Resources/charity_data.csv')
charity_df.head(10)
```

Out[2]:

	EIN	NAME	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CA
0	10520599	BLUE KNIGHTS MOTORCYCLE CLUB	T10	Independent	C1000	Product
1	10531628	AMERICAN CHESAPEAKE CLUB CHARITABLE TR	T3	Independent	C2000	Preserva
2	10547893	ST CLOUD PROFESSIONAL FIREFIGHTERS	T5	CompanySponsored	C3000	Product
3	10553066	SOUTHSIDE ATHLETIC ASSOCIATION	T3	CompanySponsored	C2000	Preserva
4	10556103	GENETIC RESEARCH INSTITUTE OF THE DESERT	T3	Independent	C1000	Heathc
5	10556855	MINORITY ORGAN & TISSUE TRANSPLANT & EDUCATION...	T3	Independent	C1200	Preserva
6	10558440	FRIENDS OF ARTS COUNCIL OF GREATER DENHAM SPRI...	T3	Independent	C1000	Preserva
7	10566033	ISRAEL EMERGENCY ALLIANCE	T3	Independent	C2000	Preserva
8	10570430	ARAMCO BRATS INC	T7	Independent	C1000	Product
9	10571689	INTERNATIONAL ASSOCIATION OF FIRE FIGHTERS	T5	CompanySponsored	C3000	Product

Data Pre-Processing

Data contains categories without numerical data and bucketing can be used to determine unique columns ideal for data analysis.

In [3]:
```
# Generate our categorical variable list
charity_cat = charity_df.dtypes[charity_df.dtypes == "object"].index.tol
ist()

# Check the number of unique values in each column
charity_df[charity_cat].nunique()
```

Out[3]:
```
NAME                      19568
APPLICATION_TYPE             17
AFFILIATION                   6
CLASSIFICATION               71
USE_CASE                      5
ORGANIZATION                  4
INCOME_AMT                    9
SPECIAL_CONSIDERATIONS        2
dtype: int64
```

Check Special Considerations Column

This column has "yes" or "no" values for if a charity is considered or not for donation

In [4]:
```
print(charity_df['SPECIAL_CONSIDERATIONS'].value_counts())
```

```
N    34272
Y       27
Name: SPECIAL_CONSIDERATIONS, dtype: int64
```

Note: Only 27 charities have yes to be considered, 34,272 not to be considered for special considerations.

In [5]:
```python
# Clean up unusuable data columns "EIN", "Status", "Name" -
# don't tell us anything about the data or contribute to model
charity_df.drop(['STATUS', 'EIN', 'NAME'], axis=1, inplace=True)
charity_df.head(10)
```

Out[5]:

	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CASE	ORGANIZATION	INCOI
0	T10	Independent	C1000	ProductDev	Association	
1	T3	Independent	C2000	Preservation	Co-operative	
2	T5	CompanySponsored	C3000	ProductDev	Association	
3	T3	CompanySponsored	C2000	Preservation	Trust	100(
4	T3	Independent	C1000	Heathcare	Trust	
5	T3	Independent	C1200	Preservation	Trust	
6	T3	Independent	C1000	Preservation	Trust	
7	T3	Independent	C2000	Preservation	Trust	1
8	T7	Independent	C1000	ProductDev	Trust	
9	T5	CompanySponsored	C3000	ProductDev	Association	

In [ ]:
```python
# Store 'Name' column in separate df to merge later
```

In [6]:
```python
# Generate our new cleaned-up categorical variable list
charity_cat_new = charity_df.dtypes[charity_df.dtypes == "object"].index
.tolist()

# Check the number of unique values in each column
charity_df[charity_cat_new].nunique()
```

Out[6]:
```
APPLICATION_TYPE          17
AFFILIATION                6
CLASSIFICATION            71
USE_CASE                   5
ORGANIZATION               4
INCOME_AMT                 9
SPECIAL_CONSIDERATIONS     2
dtype: int64
```

Bucket two largest columns (application_type and classification)

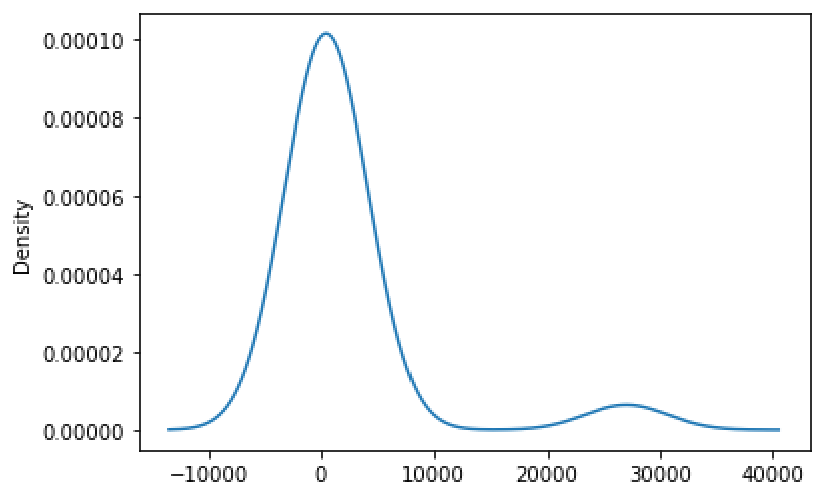
Application Types

In [7]:
```
# Check the unique value counts in APPLICATION_TYPE column
app_types = charity_df.APPLICATION_TYPE.value_counts()
app_types
```

Out[7]:
```
T3     27037
T4      1542
T6      1216
T5      1173
T19     1065
T8       737
T7       725
T10      528
T9       156
T13       66
T12       27
T2        16
T25        3
T14        3
T15        2
T29        2
T17        1
Name: APPLICATION_TYPE, dtype: int64
```

In [8]:
```
# Visualize the value counts by producing a density plot in Pandas
app_types.plot.density()
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3f92cf50>
```



According to the density plot...(finish up to configure to show better spread of data)

Bucketing Application Type Column Values

In [9]:
```
# Determine which values to replace
replace_apps = list(app_types[app_types < 200].index)
```

In [10]:
```python
# Replace in DataFrame with other any instances of < 200.
for APPLICATION_TYPE in replace_apps:
    charity_df.APPLICATION_TYPE = charity_df.APPLICATION_TYPE.replace(AP
PLICATION_TYPE,"Other")
```

In [11]:
```python
# Check to make sure bucketing was successful (applications removed)
charity_df.APPLICATION_TYPE.value_counts()
```

Out[11]:
```
T3       27037
T4        1542
T6        1216
T5        1173
T19       1065
T8         737
T7         725
T10        528
Other      276
Name: APPLICATION_TYPE, dtype: int64
```

In [12]:
```python
# Check storing of other applications not bucketed
other_app_types = app_types[app_types < 200]
other_app_types
```

Out[12]:
```
T9     156
T13     66
T12     27
T2      16
T25      3
T14      3
T15      2
T29      2
T17      1
Name: APPLICATION_TYPE, dtype: int64
```

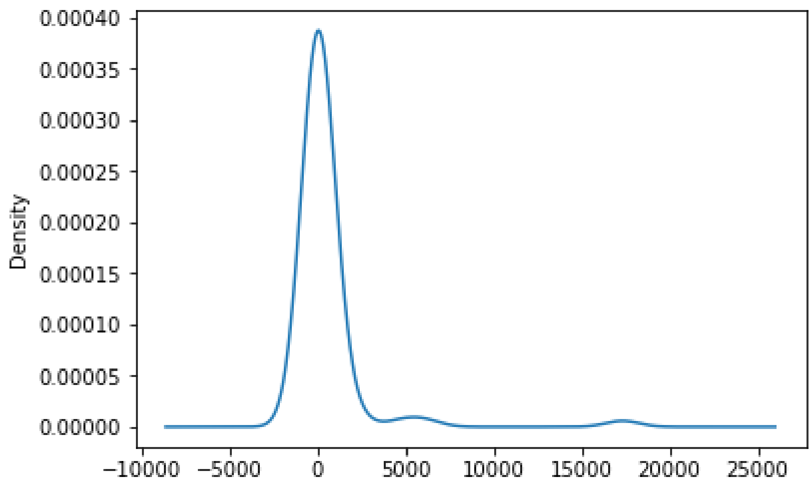
Classification Types

In [13]:
```python
# Check the unique value counts in APPLICATION_TYPE column
class_types = charity_df.CLASSIFICATION.value_counts()
class_types
```

Out[13]:
```
C1000    17326
C2000     6074
C1200     4837
C3000     1918
C2100     1883
         ...
C2570        1
C2380        1
C1283        1
C4200        1
C2600        1
Name: CLASSIFICATION, Length: 71, dtype: int64
```

In [14]:
```
# Visualize the value counts by producing a density plot in Pandas
class_types.plot.density()
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3fec1350>
```



In [15]:
```
# Check other classification types
other_class_types = class_types[class_types < 200]
other_class_types
```

Out[15]:
```
C4000    194
C5000    116
C1270    114
C2700    104
C2800     95
        ...
C2570      1
C2380      1
C1283      1
C4200      1
C2600      1
Name: CLASSIFICATION, Length: 64, dtype: int64
```

Bucketing Classification Column Values

In [16]:
```
# Determine which values to replace
replace_class = list(class_types[class_types < 200].index)
```

In [17]:
```
# Replace in DataFrame with other any instances of < 100.
for CLASSIFICATION in replace_class:
    charity_df.CLASSIFICATION = charity_df.CLASSIFICATION.replace(CLASSI
FICATION,"Other")
```

In [18]:
```
# Check to make sure bucketing was successful (applications removed)
charity_df.CLASSIFICATION.value_counts()
```

Out[18]:
```
C1000    17326
C2000     6074
C1200     4837
C3000     1918
C2100     1883
Other     1197
C7000      777
C1700      287
Name: CLASSIFICATION, dtype: int64
```

One Hot Encoding

In [19]:
```
# Create a OneHotEncoder instance
enc = OneHotEncoder(sparse=False)

# Fit and transform the OneHotEncoder using the categorical variable lis
t
encode_df = pd.DataFrame(enc.fit_transform(charity_df[charity_cat_new]))

# Add the encoded variable names to the DataFrame
encode_df.columns = enc.get_feature_names(charity_cat_new)
encode_df.head()
```

Out[19]:

	APPLICATION_TYPE_Other	APPLICATION_TYPE_T10	APPLICATION_TYPE_T19	APPLICATION_TY
0	0.0	1.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

5 rows × 43 columns

```
In [20]: # Merge one-hot encoded features and drop the originals
charity_df = charity_df.merge(encode_df,left_index=True, right_index=True)
charity_df = charity_df.drop(charity_cat_new,1)
charity_df.head()
```

Out[20]:

	ASK_AMT	IS_SUCCESSFUL	APPLICATION_TYPE_Other	APPLICATION_TYPE_T10	APPLICATION
0	5000	1	0.0	1.0	
1	108590	1	0.0	0.0	
2	5000	0	0.0	0.0	
3	6692	1	0.0	0.0	
4	142590	1	0.0	0.0	

5 rows × 45 columns

Training and Testing Datasets

```
In [21]: # Create variables for feature and target out of pre-processed data
# Feature (all other categories), target ("IS_SUCCESSFUL")
y = charity_df["IS_SUCCESSFUL"].values
X = charity_df.drop(["IS_SUCCESSFUL"],1).values

# Split training/test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=78)
```

Standardize Dataset

```
In [22]: # Create a StandardScaler instance
scaler = StandardScaler()

# Fit the StandardScaler
X_scaler = scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```
In [23]: X_train_scaled.shape
```

```
Out[23]: (25724, 44)
```

```
In [24]: X_test_scaled.shape
```

```
Out[24]: (8575, 44)
```

Instantiate and Evaluate Random Forest Classifier

In [25]:
```
# Create a random forest classifier.
rf_model = RandomForestClassifier(n_estimators=128, random_state=78)

# Fitting the model
rf_model = rf_model.fit(X_train_scaled, y_train)

# Evaluate the model
y_pred = rf_model.predict(X_test_scaled)
print(f" Random forest predictive accuracy: {accuracy_score(y_test,y_pred):.3f}")
```

```
 Random forest predictive accuracy: 0.710
```

Summary:

Random forest only predicts at 70% accuracy, need to achieve an accuracy of higher than 75%

Deep Learning Model #1 - Using input features (44 neurons, half that for second layer 22 neurons), epoch = 50

In [26]:
```
# Define the model - deep neural net
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  44
hidden_nodes_layer2 = 22

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_in
put_features, activation="relu")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="rel
u"))


# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
WARNING:tensorflow:From /opt/anaconda3/envs/mlenv/lib/python3.7/site-pa
ckages/tensorflow/python/ops/init_ops.py:1251: calling VarianceScaling.
__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated
and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it
to the constructor
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 44)                1980
_________________________________________________________________
dense_1 (Dense)              (None, 22)                990
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 23
=================================================================
Total params: 2,993
Trainable params: 2,993
Non-trainable params: 0
_________________________________________________________________
```

In [27]:
```
# Compile the Sequential model together and customize metrics
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accur
acy"])
```

```
WARNING:tensorflow:From /opt/anaconda3/envs/mlenv/lib/python3.7/site-pa
ckages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_support.<loca
ls>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and wi
ll be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [28]:
```
# Train the model
fit_model = nn.fit(X_train_scaled, y_train, epochs=50)
```

```
Epoch 1/50
25724/25724 [==============================] - 1s 49us/sample - loss:
0.5724 - acc: 0.7199
Epoch 2/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5545 - acc: 0.7311
Epoch 3/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5518 - acc: 0.7312
Epoch 4/50
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5493 - acc: 0.7307
Epoch 5/50
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5483 - acc: 0.7332
Epoch 6/50
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5472 - acc: 0.7339
Epoch 7/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5467 - acc: 0.7324
Epoch 8/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5459 - acc: 0.7342
Epoch 9/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5454 - acc: 0.7337
Epoch 10/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5454 - acc: 0.7342
Epoch 11/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5446 - acc: 0.7346
Epoch 12/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5442 - acc: 0.7347
Epoch 13/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5442 - acc: 0.7341
Epoch 14/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5437 - acc: 0.7363
Epoch 15/50
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5433 - acc: 0.7344
Epoch 16/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5434 - acc: 0.7358
Epoch 17/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5431 - acc: 0.7361
Epoch 18/50
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5422 - acc: 0.7362
Epoch 19/50
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5426 - acc: 0.7364
```

```
Epoch 20/50
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5422 - acc: 0.7365
Epoch 21/50
25724/25724 [==============================] - 1s 54us/sample - loss:
0.5418 - acc: 0.7362
Epoch 22/50
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5414 - acc: 0.7379
Epoch 23/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5414 - acc: 0.7359
Epoch 24/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5418 - acc: 0.7360
Epoch 25/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5409 - acc: 0.7364
Epoch 26/50
25724/25724 [==============================] - 1s 52us/sample - loss:
0.5411 - acc: 0.73800s - loss: 0.5407 - acc: 0.7
Epoch 27/50
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5404 - acc: 0.7364
Epoch 28/50
25724/25724 [==============================] - 1s 51us/sample - loss:
0.5409 - acc: 0.7378
Epoch 29/50
25724/25724 [==============================] - 1s 52us/sample - loss:
0.5404 - acc: 0.7378
Epoch 30/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5404 - acc: 0.7377
Epoch 31/50
25724/25724 [==============================] - 1s 51us/sample - loss:
0.5401 - acc: 0.7379
Epoch 32/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5402 - acc: 0.7380
Epoch 33/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5399 - acc: 0.7389
Epoch 34/50
25724/25724 [==============================] - 2s 60us/sample - loss:
0.5405 - acc: 0.7370
Epoch 35/50
25724/25724 [==============================] - 1s 51us/sample - loss:
0.5395 - acc: 0.7388
Epoch 36/50
25724/25724 [==============================] - 1s 52us/sample - loss:
0.5394 - acc: 0.7389
Epoch 37/50
25724/25724 [==============================] - 1s 55us/sample - loss:
0.5394 - acc: 0.7392
Epoch 38/50
25724/25724 [==============================] - 1s 52us/sample - loss:
0.5393 - acc: 0.7384
```

```
Epoch 39/50
25724/25724 [==============================] - 1s 47us/sample - loss: 0.5390 - acc: 0.7385
Epoch 40/50
25724/25724 [==============================] - 1s 49us/sample - loss: 0.5395 - acc: 0.7388
Epoch 41/50
25724/25724 [==============================] - 1s 45us/sample - loss: 0.5387 - acc: 0.7371
Epoch 42/50
25724/25724 [==============================] - 1s 43us/sample - loss: 0.5391 - acc: 0.7392
Epoch 43/50
25724/25724 [==============================] - 1s 44us/sample - loss: 0.5388 - acc: 0.7385
Epoch 44/50
25724/25724 [==============================] - 1s 43us/sample - loss: 0.5390 - acc: 0.7382
Epoch 45/50
25724/25724 [==============================] - 1s 45us/sample - loss: 0.5383 - acc: 0.7392
Epoch 46/50
25724/25724 [==============================] - 1s 51us/sample - loss: 0.5383 - acc: 0.7399
Epoch 47/50
25724/25724 [==============================] - 1s 50us/sample - loss: 0.5385 - acc: 0.73891s - l
Epoch 48/50
25724/25724 [==============================] - 1s 47us/sample - loss: 0.5380 - acc: 0.73930s - loss: 0.5368 - acc: 0.
Epoch 49/50
25724/25724 [==============================] - 1s 56us/sample - loss: 0.5384 - acc: 0.7392
Epoch 50/50
25724/25724 [==============================] - 1s 46us/sample - loss: 0.5377 - acc: 0.7393
```

In [29]:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
8575/8575 - 0s - loss: 0.5542 - acc: 0.7264
Loss: 0.5541565223546487, Accuracy: 0.7264139652252197
```

Deep Learning Model #2 - Same neurons, compared to first model (2 hidden layers 44 neurons, 24 neurons), epoch = 100

In [30]:

```
# Define the model - deep neural net
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  44
hidden_nodes_layer2 = 22

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_in
put_features, activation="relu")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="rel
u"))


# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_3 (Dense)              (None, 44)                1980      
_________________________________________________________________
dense_4 (Dense)              (None, 22)                990       
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 23        
=================================================================
Total params: 2,993
Trainable params: 2,993
Non-trainable params: 0
_________________________________________________________________
```

In [31]:

```
# Compile the Sequential model together and customize metrics
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accur
acy"])
```

In [32]:
```
# Train the model
fit_model = nn.fit(X_train_scaled, y_train, epochs=100)
```

```
Epoch 1/100
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5742 - acc: 0.7156
Epoch 2/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5545 - acc: 0.7284
Epoch 3/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5515 - acc: 0.7306
Epoch 4/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5498 - acc: 0.7307
Epoch 5/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5478 - acc: 0.7314
Epoch 6/100
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5474 - acc: 0.7325
Epoch 7/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5463 - acc: 0.7331
Epoch 8/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5458 - acc: 0.7335
Epoch 9/100
25724/25724 [==============================] - 1s 38us/sample - loss:
0.5449 - acc: 0.7341
Epoch 10/100
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5443 - acc: 0.7331
Epoch 11/100
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5440 - acc: 0.7360
Epoch 12/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5434 - acc: 0.7360
Epoch 13/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5429 - acc: 0.7356
Epoch 14/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5434 - acc: 0.7361
Epoch 15/100
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5429 - acc: 0.7360
Epoch 16/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5429 - acc: 0.7363
Epoch 17/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5424 - acc: 0.7355
Epoch 18/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5425 - acc: 0.7371
Epoch 19/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5415 - acc: 0.7383
```

```
Epoch 20/100
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5414 - acc: 0.7371
Epoch 21/100
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5410 - acc: 0.7376
Epoch 22/100
25724/25724 [==============================] - 1s 51us/sample - loss:
0.5407 - acc: 0.7371
Epoch 23/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5413 - acc: 0.73781s - loss: 0.5444 - - ETA: 0s - loss: 0.5422 - acc
Epoch 24/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5409 - acc: 0.7374
Epoch 25/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5405 - acc: 0.7380
Epoch 26/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5403 - acc: 0.7389
Epoch 27/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5400 - acc: 0.7389
Epoch 28/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5402 - acc: 0.7378
Epoch 29/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5400 - acc: 0.7373
Epoch 30/100
25724/25724 [==============================] - 1s 51us/sample - loss:
0.5395 - acc: 0.73890s - loss: 0.5391 - a
Epoch 31/100
25724/25724 [==============================] - 1s 52us/sample - loss:
0.5396 - acc: 0.7380
Epoch 32/100
25724/25724 [==============================] - 1s 49us/sample - loss:
0.5394 - acc: 0.7389
Epoch 33/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5395 - acc: 0.7387
Epoch 34/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5388 - acc: 0.7389
Epoch 35/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5387 - acc: 0.7394
Epoch 36/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5391 - acc: 0.7392
Epoch 37/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5389 - acc: 0.7390
Epoch 38/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5385 - acc: 0.7385
```

```
Epoch 39/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5381 - acc: 0.7387
Epoch 40/100
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5383 - acc: 0.7385
Epoch 41/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5385 - acc: 0.7399
Epoch 42/100
25724/25724 [==============================] - 1s 38us/sample - loss:
0.5384 - acc: 0.7394
Epoch 43/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5383 - acc: 0.7395
Epoch 44/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5382 - acc: 0.7395
Epoch 45/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5380 - acc: 0.7400
Epoch 46/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5380 - acc: 0.7397
Epoch 47/100
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5380 - acc: 0.7399
Epoch 48/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5375 - acc: 0.7406
Epoch 49/100
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5374 - acc: 0.7395
Epoch 50/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5378 - acc: 0.7382
Epoch 51/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5377 - acc: 0.7399
Epoch 52/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5373 - acc: 0.7393
Epoch 53/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5371 - acc: 0.7400
Epoch 54/100
25724/25724 [==============================] - 1s 38us/sample - loss:
0.5372 - acc: 0.7399
Epoch 55/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5374 - acc: 0.7387
Epoch 56/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5372 - acc: 0.7388
Epoch 57/100
25724/25724 [==============================] - 1s 49us/sample - loss:
0.5370 - acc: 0.7403
```

```
Epoch 58/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5372 - acc: 0.73990s - loss: 0
Epoch 59/100
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5368 - acc: 0.7401
Epoch 60/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5367 - acc: 0.7395
Epoch 61/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5367 - acc: 0.7405
Epoch 62/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5366 - acc: 0.74040s - loss: 0.540
Epoch 63/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5364 - acc: 0.7395
Epoch 64/100
25724/25724 [==============================] - 1s 39us/sample - loss:
0.5367 - acc: 0.7391
Epoch 65/100
25724/25724 [==============================] - 1s 54us/sample - loss:
0.5365 - acc: 0.7406
Epoch 66/100
25724/25724 [==============================] - 2s 76us/sample - loss:
0.5363 - acc: 0.7408
Epoch 67/100
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5366 - acc: 0.7399
Epoch 68/100
25724/25724 [==============================] - 2s 81us/sample - loss:
0.5359 - acc: 0.7407
Epoch 69/100
25724/25724 [==============================] - 3s 110us/sample - loss:
0.5363 - acc: 0.7401
Epoch 70/100
25724/25724 [==============================] - 2s 60us/sample - loss:
0.5362 - acc: 0.7406
Epoch 71/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5361 - acc: 0.74030s - loss: 0.5343 -
Epoch 72/100
25724/25724 [==============================] - 1s 56us/sample - loss:
0.5363 - acc: 0.7404
Epoch 73/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5357 - acc: 0.7394
Epoch 74/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5356 - acc: 0.7403
Epoch 75/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5358 - acc: 0.74070s - loss: 0.5370 - acc: 0
Epoch 76/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5357 - acc: 0.7408
```

```
Epoch 77/100
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5357 - acc: 0.7408
Epoch 78/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5357 - acc: 0.7410
Epoch 79/100
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5358 - acc: 0.7403
Epoch 80/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5354 - acc: 0.7401
Epoch 81/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5354 - acc: 0.7399
Epoch 82/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5352 - acc: 0.7408
Epoch 83/100
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5357 - acc: 0.7416
Epoch 84/100
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5355 - acc: 0.7414
Epoch 85/100
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5355 - acc: 0.7406
Epoch 86/100
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5349 - acc: 0.7410
Epoch 87/100
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5353 - acc: 0.7406
Epoch 88/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5351 - acc: 0.7408
Epoch 89/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5352 - acc: 0.7406
Epoch 90/100
25724/25724 [==============================] - 1s 54us/sample - loss:
0.5350 - acc: 0.7411
Epoch 91/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5351 - acc: 0.7416
Epoch 92/100
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5349 - acc: 0.7417
Epoch 93/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5350 - acc: 0.7400
Epoch 94/100
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5349 - acc: 0.7411
Epoch 95/100
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5348 - acc: 0.7409
```

```
Epoch 96/100
25724/25724 [==============================] - 1s 50us/sample - loss: 
0.5351 - acc: 0.7408
Epoch 97/100
25724/25724 [==============================] - 1s 58us/sample - loss: 
0.5347 - acc: 0.74180s - los
Epoch 98/100
25724/25724 [==============================] - 1s 55us/sample - loss: 
0.5348 - acc: 0.7419
Epoch 99/100
25724/25724 [==============================] - 1s 53us/sample - loss: 
0.5347 - acc: 0.7404
Epoch 100/100
25724/25724 [==============================] - 1s 55us/sample - loss: 
0.5348 - acc: 0.7408
```

In [33]:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
8575/8575 - 0s - loss: 0.5557 - acc: 0.7248
Loss: 0.5557105709859999, Accuracy: 0.724781334400177
```

Deep Learning Model #3 - Using practice module example of 2 hidden layers (44 neurons, 22 neurons), epoch = 50, activation first hidden layer = tanh

In [37]:
```
# Define the model - deep neural net
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  44
hidden_nodes_layer2 = 22

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_in
put_features, activation="tanh")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="rel
u"))


# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_9 (Dense)              (None, 44)                1980      
_________________________________________________________________
dense_10 (Dense)             (None, 22)                990       
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 23        
=================================================================
Total params: 2,993
Trainable params: 2,993
Non-trainable params: 0
_________________________________________________________________
```

In [38]:
```
# Compile the Sequential model together and customize metrics
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accur
acy"])
```

In [39]:
```
# Train the model
fit_model = nn.fit(X_train_scaled, y_train, epochs=50)
```

```
Epoch 1/50
25724/25724 [==============================] - 2s 63us/sample - loss: 
0.5697 - acc: 0.7201
Epoch 2/50
25724/25724 [==============================] - 1s 42us/sample - loss: 
0.5541 - acc: 0.7280
Epoch 3/50
25724/25724 [==============================] - 1s 43us/sample - loss: 
0.5510 - acc: 0.7318
Epoch 4/50
25724/25724 [==============================] - 1s 41us/sample - loss: 
0.5497 - acc: 0.7308
Epoch 5/50
25724/25724 [==============================] - 1s 40us/sample - loss: 
0.5481 - acc: 0.7332
Epoch 6/50
25724/25724 [==============================] - 1s 40us/sample - loss: 
0.5476 - acc: 0.7328
Epoch 7/50
25724/25724 [==============================] - 1s 40us/sample - loss: 
0.5466 - acc: 0.7334
Epoch 8/50
25724/25724 [==============================] - 1s 45us/sample - loss: 
0.5461 - acc: 0.7338
Epoch 9/50
25724/25724 [==============================] - 1s 44us/sample - loss: 
0.5454 - acc: 0.7342
Epoch 10/50
25724/25724 [==============================] - 1s 41us/sample - loss: 
0.5450 - acc: 0.7341
Epoch 11/50
25724/25724 [==============================] - 1s 52us/sample - loss: 
0.5446 - acc: 0.7356
Epoch 12/50
25724/25724 [==============================] - 1s 43us/sample - loss: 
0.5440 - acc: 0.7344
Epoch 13/50
25724/25724 [==============================] - 1s 40us/sample - loss: 
0.5437 - acc: 0.7342
Epoch 14/50
25724/25724 [==============================] - 1s 40us/sample - loss: 
0.5443 - acc: 0.7353
Epoch 15/50
25724/25724 [==============================] - 1s 41us/sample - loss: 
0.5433 - acc: 0.7345
Epoch 16/50
25724/25724 [==============================] - 1s 54us/sample - loss: 
0.5436 - acc: 0.7356
Epoch 17/50
25724/25724 [==============================] - 1s 46us/sample - loss: 
0.5427 - acc: 0.7356
Epoch 18/50
25724/25724 [==============================] - 1s 45us/sample - loss: 
0.5431 - acc: 0.7361
Epoch 19/50
25724/25724 [==============================] - 1s 41us/sample - loss: 
0.5423 - acc: 0.7364
```

```
Epoch 20/50
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5422 - acc: 0.7370
Epoch 21/50
25724/25724 [==============================] - 1s 41us/sample - loss:
0.5415 - acc: 0.7374
Epoch 22/50
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5420 - acc: 0.7367
Epoch 23/50
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5417 - acc: 0.73650s - loss: 0.5410 - acc: 0.737
Epoch 24/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5409 - acc: 0.7368
Epoch 25/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5409 - acc: 0.7378
Epoch 26/50
25724/25724 [==============================] - 1s 40us/sample - loss:
0.5413 - acc: 0.7365
Epoch 27/50
25724/25724 [==============================] - 1s 42us/sample - loss:
0.5409 - acc: 0.7368
Epoch 28/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5409 - acc: 0.7376
Epoch 29/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5406 - acc: 0.7383
Epoch 30/50
25724/25724 [==============================] - 1s 44us/sample - loss:
0.5407 - acc: 0.7376
Epoch 31/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5405 - acc: 0.7379
Epoch 32/50
25724/25724 [==============================] - 1s 53us/sample - loss:
0.5399 - acc: 0.7375
Epoch 33/50
25724/25724 [==============================] - 1s 55us/sample - loss:
0.5399 - acc: 0.7379
Epoch 34/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5401 - acc: 0.73841s - loss:
Epoch 35/50
25724/25724 [==============================] - 1s 43us/sample - loss:
0.5396 - acc: 0.7395
Epoch 36/50
25724/25724 [==============================] - 1s 49us/sample - loss:
0.5400 - acc: 0.7380
Epoch 37/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5393 - acc: 0.7391
Epoch 38/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5392 - acc: 0.7385
```

```
Epoch 39/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5396 - acc: 0.7391
Epoch 40/50
25724/25724 [==============================] - 1s 48us/sample - loss:
0.5390 - acc: 0.7387
Epoch 41/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5393 - acc: 0.7382
Epoch 42/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5389 - acc: 0.7399
Epoch 43/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5387 - acc: 0.7386
Epoch 44/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5385 - acc: 0.7390
Epoch 45/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5386 - acc: 0.7393
Epoch 46/50
25724/25724 [==============================] - 1s 50us/sample - loss:
0.5383 - acc: 0.7397
Epoch 47/50
25724/25724 [==============================] - 1s 47us/sample - loss:
0.5383 - acc: 0.7388
Epoch 48/50
25724/25724 [==============================] - 1s 45us/sample - loss:
0.5386 - acc: 0.7397
Epoch 49/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5382 - acc: 0.7404
Epoch 50/50
25724/25724 [==============================] - 1s 46us/sample - loss:
0.5383 - acc: 0.7390
```

In [40]:
```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
8575/8575 - 0s - loss: 0.5514 - acc: 0.7280
Loss: 0.5514009420378215, Accuracy: 0.7280466556549072
```

Deep Learning Model #4 - Using practice module example of 3 hidden layers, less neurons (44 neurons, 32 neurons), epoch = 50, tanh, relu

In [129]:
```
# Define the model - deep neural net
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  44
hidden_nodes_layer2 = 32

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_in
put_features, activation="tanh")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, input_dim=number
_input_features, activation="relu"))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_35"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_107 (Dense)            (None, 44)                1980
_________________________________________________________________
dense_108 (Dense)            (None, 32)                1440
_________________________________________________________________
dense_109 (Dense)            (None, 1)                 33
=================================================================
Total params: 3,453
Trainable params: 3,453
Non-trainable params: 0
_________________________________________________________________
```

In [130]:
```
# Compile the Sequential model together and customize metrics
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accur
acy"])
```

In [131]:

```
# Train the model
fit_model = nn.fit(X_train_scaled, y_train, epochs=50)
```

```
Epoch 1/50
25724/25724 [==============================] - 2s 82us/sample - loss:
0.5687 - acc: 0.7222
Epoch 2/50
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5549 - acc: 0.7287
Epoch 3/50
25724/25724 [==============================] - 2s 66us/sample - loss:
0.5521 - acc: 0.7292
Epoch 4/50
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5499 - acc: 0.7322
Epoch 5/50
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5488 - acc: 0.73240s - loss: 0.5487 - acc:
Epoch 6/50
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5475 - acc: 0.7325
Epoch 7/50
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5475 - acc: 0.7329
Epoch 8/50
25724/25724 [==============================] - 2s 61us/sample - loss:
0.5461 - acc: 0.7339
Epoch 9/50
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5458 - acc: 0.7335
Epoch 10/50
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5458 - acc: 0.7344
Epoch 11/50
25724/25724 [==============================] - 2s 61us/sample - loss:
0.5445 - acc: 0.7343
Epoch 12/50
25724/25724 [==============================] - 2s 62us/sample - loss:
0.5447 - acc: 0.7352
Epoch 13/50
25724/25724 [==============================] - 2s 66us/sample - loss:
0.5443 - acc: 0.7359
Epoch 14/50
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5439 - acc: 0.7353
Epoch 15/50
25724/25724 [==============================] - 2s 76us/sample - loss:
0.5436 - acc: 0.7345
Epoch 16/50
25724/25724 [==============================] - 2s 66us/sample - loss:
0.5431 - acc: 0.7357
Epoch 17/50
25724/25724 [==============================] - 2s 61us/sample - loss:
0.5432 - acc: 0.7354
Epoch 18/50
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5424 - acc: 0.7358
Epoch 19/50
25724/25724 [==============================] - 2s 59us/sample - loss:
0.5429 - acc: 0.7364
```

```
Epoch 20/50
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5421 - acc: 0.7357
Epoch 21/50
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5421 - acc: 0.7362
Epoch 22/50
25724/25724 [==============================] - 2s 61us/sample - loss:
0.5422 - acc: 0.7362
Epoch 23/50
25724/25724 [==============================] - 2s 75us/sample - loss:
0.5416 - acc: 0.73710s - loss: 0.5426 -
Epoch 24/50
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5417 - acc: 0.7360
Epoch 25/50
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5413 - acc: 0.7368
Epoch 26/50
25724/25724 [==============================] - 2s 62us/sample - loss:
0.5409 - acc: 0.7369
Epoch 27/50
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5413 - acc: 0.7377
Epoch 28/50
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5411 - acc: 0.7371
Epoch 29/50
25724/25724 [==============================] - 2s 61us/sample - loss:
0.5406 - acc: 0.7387
Epoch 30/50
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5403 - acc: 0.7376
Epoch 31/50
25724/25724 [==============================] - 2s 71us/sample - loss:
0.5402 - acc: 0.7385
Epoch 32/50
25724/25724 [==============================] - 2s 64us/sample - loss:
0.5404 - acc: 0.7369
Epoch 33/50
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5401 - acc: 0.7381
Epoch 34/50
25724/25724 [==============================] - 2s 70us/sample - loss:
0.5397 - acc: 0.7373
Epoch 35/50
25724/25724 [==============================] - 2s 75us/sample - loss:
0.5398 - acc: 0.7379
Epoch 36/50
25724/25724 [==============================] - 2s 73us/sample - loss:
0.5396 - acc: 0.7378
Epoch 37/50
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5393 - acc: 0.7384
Epoch 38/50
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5393 - acc: 0.7388
```

```
Epoch 39/50
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5395 - acc: 0.7380
Epoch 40/50
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5392 - acc: 0.7376
Epoch 41/50
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5387 - acc: 0.7393
Epoch 42/50
25724/25724 [==============================] - 2s 83us/sample - loss:
0.5391 - acc: 0.7377
Epoch 43/50
25724/25724 [==============================] - 2s 79us/sample - loss:
0.5384 - acc: 0.73780s - loss: 0.5372 -
Epoch 44/50
25724/25724 [==============================] - 2s 81us/sample - loss:
0.5386 - acc: 0.7379
Epoch 45/50
25724/25724 [==============================] - 2s 75us/sample - loss:
0.5387 - acc: 0.7391
Epoch 46/50
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5383 - acc: 0.7386
Epoch 47/50
25724/25724 [==============================] - 2s 77us/sample - loss:
0.5385 - acc: 0.7382
Epoch 48/50
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5383 - acc: 0.7384
Epoch 49/50
25724/25724 [==============================] - 2s 64us/sample - loss:
0.5382 - acc: 0.7392
Epoch 50/50
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5379 - acc: 0.7381
```

In [132]:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
8575/8575 - 0s - loss: 0.5499 - acc: 0.7278
Loss: 0.5499131618505316, Accuracy: 0.7278134226799011
```

Deep Learning Model #5 - Boost epoch, accuracy best at 2 hidden layers (44 neurons, 32 neurons), epoch = 100

In [142]:

```
# Define the model - deep neural net
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  44
hidden_nodes_layer2 = 32

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_in
put_features, activation="tanh")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, input_dim=number
_input_features, activation="relu"))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_39"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_119 (Dense)            (None, 44)                1980
_________________________________________________________________
dense_120 (Dense)            (None, 32)                1440
_________________________________________________________________
dense_121 (Dense)            (None, 1)                 33
=================================================================
Total params: 3,453
Trainable params: 3,453
Non-trainable params: 0
_________________________________________________________________
```

In [140]:

```
# Compile the Sequential model together and customize metrics
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accur
acy"])
```

In [141]:

```
#### Train the model
fit_model = nn.fit(X_train_scaled, y_train, epochs=100)
```

```
Epoch 1/100
25724/25724 [==============================] - 2s 77us/sample - loss: 
0.5693 - acc: 0.7202
Epoch 2/100
25724/25724 [==============================] - 2s 71us/sample - loss: 
0.5545 - acc: 0.7303
Epoch 3/100
25724/25724 [==============================] - 2s 75us/sample - loss: 
0.5517 - acc: 0.7320
Epoch 4/100
25724/25724 [==============================] - 2s 66us/sample - loss: 
0.5500 - acc: 0.7324
Epoch 5/100
25724/25724 [==============================] - 2s 74us/sample - loss: 
0.5486 - acc: 0.7315
Epoch 6/100
25724/25724 [==============================] - 2s 82us/sample - loss: 
0.5482 - acc: 0.7333
Epoch 7/100
25724/25724 [==============================] - 2s 79us/sample - loss: 
0.5475 - acc: 0.7343
Epoch 8/100
25724/25724 [==============================] - 2s 89us/sample - loss: 
0.5464 - acc: 0.7334
Epoch 9/100
25724/25724 [==============================] - 2s 70us/sample - loss: 
0.5453 - acc: 0.7349
Epoch 10/100
25724/25724 [==============================] - 2s 64us/sample - loss: 
0.5454 - acc: 0.7340
Epoch 11/100
25724/25724 [==============================] - 2s 67us/sample - loss: 
0.5449 - acc: 0.7345
Epoch 12/100
25724/25724 [==============================] - 2s 66us/sample - loss: 
0.5445 - acc: 0.7366
Epoch 13/100
25724/25724 [==============================] - 2s 65us/sample - loss: 
0.5441 - acc: 0.7360
Epoch 14/100
25724/25724 [==============================] - 1s 57us/sample - loss: 
0.5441 - acc: 0.7369
Epoch 15/100
25724/25724 [==============================] - 2s 59us/sample - loss: 
0.5439 - acc: 0.7358
Epoch 16/100
25724/25724 [==============================] - 2s 70us/sample - loss: 
0.5434 - acc: 0.7354
Epoch 17/100
25724/25724 [==============================] - 1s 57us/sample - loss: 
0.5426 - acc: 0.7350
Epoch 18/100
25724/25724 [==============================] - 1s 56us/sample - loss: 
0.5429 - acc: 0.7375
Epoch 19/100
25724/25724 [==============================] - 1s 54us/sample - loss: 
0.5426 - acc: 0.7368
```

```
Epoch 20/100
25724/25724 [==============================] - 1s 57us/sample - loss:
0.5424 - acc: 0.7357
Epoch 21/100
25724/25724 [==============================] - 2s 59us/sample - loss:
0.5421 - acc: 0.7358
Epoch 22/100
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5417 - acc: 0.7368
Epoch 23/100
25724/25724 [==============================] - 2s 95us/sample - loss:
0.5418 - acc: 0.7383
Epoch 24/100
25724/25724 [==============================] - 2s 80us/sample - loss:
0.5415 - acc: 0.7377
Epoch 25/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5414 - acc: 0.7380
Epoch 26/100
25724/25724 [==============================] - 2s 64us/sample - loss:
0.5411 - acc: 0.7373
Epoch 27/100
25724/25724 [==============================] - 2s 73us/sample - loss:
0.5411 - acc: 0.7390
Epoch 28/100
25724/25724 [==============================] - 2s 83us/sample - loss:
0.5406 - acc: 0.73730s - loss: 0.5406 - acc: 0.737
Epoch 29/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5402 - acc: 0.73790s - loss: 0.
Epoch 30/100
25724/25724 [==============================] - 2s 71us/sample - loss:
0.5403 - acc: 0.7385
Epoch 31/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5403 - acc: 0.7376
Epoch 32/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5400 - acc: 0.7386
Epoch 33/100
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5400 - acc: 0.7385
Epoch 34/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5400 - acc: 0.7384
Epoch 35/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5395 - acc: 0.7388
Epoch 36/100
25724/25724 [==============================] - 2s 75us/sample - loss:
0.5399 - acc: 0.7381
Epoch 37/100
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5394 - acc: 0.7385
Epoch 38/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5391 - acc: 0.7385
```

```
Epoch 39/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5389 - acc: 0.7385
Epoch 40/100
25724/25724 [==============================] - 2s 75us/sample - loss:
0.5388 - acc: 0.7393
Epoch 41/100
25724/25724 [==============================] - 2s 73us/sample - loss:
0.5386 - acc: 0.7383
Epoch 42/100
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5388 - acc: 0.7397
Epoch 43/100
25724/25724 [==============================] - 2s 78us/sample - loss:
0.5383 - acc: 0.7395
Epoch 44/100
25724/25724 [==============================] - 2s 78us/sample - loss:
0.5381 - acc: 0.7383
Epoch 45/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5383 - acc: 0.7392
Epoch 46/100
25724/25724 [==============================] - 2s 70us/sample - loss:
0.5379 - acc: 0.7397
Epoch 47/100
25724/25724 [==============================] - 2s 81us/sample - loss:
0.5385 - acc: 0.7390
Epoch 48/100
25724/25724 [==============================] - 2s 81us/sample - loss:
0.5382 - acc: 0.7383
Epoch 49/100
25724/25724 [==============================] - 2s 70us/sample - loss:
0.5380 - acc: 0.7396
Epoch 50/100
25724/25724 [==============================] - 2s 81us/sample - loss:
0.5375 - acc: 0.7392
Epoch 51/100
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5377 - acc: 0.7392
Epoch 52/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5376 - acc: 0.7390
Epoch 53/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5373 - acc: 0.7394
Epoch 54/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5372 - acc: 0.7400
Epoch 55/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5370 - acc: 0.7397
Epoch 56/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5370 - acc: 0.7400
Epoch 57/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5368 - acc: 0.7396
```

```
Epoch 58/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5370 - acc: 0.7397
Epoch 59/100
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5368 - acc: 0.7399
Epoch 60/100
25724/25724 [==============================] - 2s 62us/sample - loss:
0.5362 - acc: 0.7398
Epoch 61/100
25724/25724 [==============================] - 2s 64us/sample - loss:
0.5364 - acc: 0.7402
Epoch 62/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5360 - acc: 0.7404
Epoch 63/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5358 - acc: 0.7395
Epoch 64/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5362 - acc: 0.7393
Epoch 65/100
25724/25724 [==============================] - 2s 63us/sample - loss:
0.5362 - acc: 0.74060s - loss: 0.5368 -
Epoch 66/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5362 - acc: 0.7407
Epoch 67/100
25724/25724 [==============================] - 2s 70us/sample - loss:
0.5361 - acc: 0.7406
Epoch 68/100
25724/25724 [==============================] - 2s 77us/sample - loss:
0.5358 - acc: 0.7406
Epoch 69/100
25724/25724 [==============================] - 2s 80us/sample - loss:
0.5359 - acc: 0.7400
Epoch 70/100
25724/25724 [==============================] - 2s 79us/sample - loss:
0.5353 - acc: 0.7407
Epoch 71/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5356 - acc: 0.7404
Epoch 72/100
25724/25724 [==============================] - 2s 73us/sample - loss:
0.5351 - acc: 0.7405
Epoch 73/100
25724/25724 [==============================] - 2s 74us/sample - loss:
0.5351 - acc: 0.7410
Epoch 74/100
25724/25724 [==============================] - 2s 64us/sample - loss:
0.5354 - acc: 0.7403
Epoch 75/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5350 - acc: 0.7406
Epoch 76/100
25724/25724 [==============================] - 2s 82us/sample - loss:
0.5351 - acc: 0.7406
```

```
Epoch 77/100
25724/25724 [==============================] - 2s 77us/sample - loss:
0.5354 - acc: 0.74080s - loss: 0.5
Epoch 78/100
25724/25724 [==============================] - 2s 77us/sample - loss:
0.5350 - acc: 0.7418
Epoch 79/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5354 - acc: 0.7400
Epoch 80/100
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5350 - acc: 0.7409
Epoch 81/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5349 - acc: 0.7405
Epoch 82/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5349 - acc: 0.7399
Epoch 83/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5348 - acc: 0.7404
Epoch 84/100
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5349 - acc: 0.7408
Epoch 85/100
25724/25724 [==============================] - 2s 68us/sample - loss:
0.5348 - acc: 0.7407
Epoch 86/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5346 - acc: 0.7412
Epoch 87/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5348 - acc: 0.7406
Epoch 88/100
25724/25724 [==============================] - 2s 67us/sample - loss:
0.5346 - acc: 0.74130s - loss: 0.537
Epoch 89/100
25724/25724 [==============================] - 2s 72us/sample - loss:
0.5341 - acc: 0.7412
Epoch 90/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5344 - acc: 0.7413
Epoch 91/100
25724/25724 [==============================] - 2s 69us/sample - loss:
0.5343 - acc: 0.7412
Epoch 92/100
25724/25724 [==============================] - 2s 76us/sample - loss:
0.5340 - acc: 0.7413
Epoch 93/100
25724/25724 [==============================] - 2s 65us/sample - loss:
0.5340 - acc: 0.7407
Epoch 94/100
25724/25724 [==============================] - 2s 70us/sample - loss:
0.5342 - acc: 0.7408
Epoch 95/100
25724/25724 [==============================] - 2s 79us/sample - loss:
0.5338 - acc: 0.7409
```

```
Epoch 96/100
25724/25724 [==============================] - 2s 76us/sample - loss:
0.5339 - acc: 0.7404
Epoch 97/100
25724/25724 [==============================] - 3s 126us/sample - loss:
0.5339 - acc: 0.7402
Epoch 98/100
25724/25724 [==============================] - 3s 112us/sample - loss:
0.5337 - acc: 0.7407
Epoch 99/100
25724/25724 [==============================] - 2s 86us/sample - loss:
0.5336 - acc: 0.7413
Epoch 100/100
25724/25724 [==============================] - 2s 66us/sample - loss:
0.5336 - acc: 0.7418
```

In [64]:

```python
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
8575/8575 - 0s - loss: 0.5519 - acc: 0.7249
Loss: 0.5518519495527529, Accuracy: 0.7248979806900024
```

In [ ]:

```python

```