project_3

September 26, 2021

1 Classification Project

In this project you will apply what you have learned about classification and TensorFlow to complete a project from Kaggle. The challenge is to achieve a high accuracy score while trying to predict which passengers survived the Titanic ship crash. After building your model, you will upload your predictions to Kaggle and submit the score that you get.

1.1 Team Members

1. Jose Martinez
2. Wren Priest
3. Maria Quintero

1.2 The Titanic Dataset

Kaggle has a dataset containing the passenger list on the Titanic. The data contains passenger features such as age, gender, ticket class, as well as whether or not they survived.

Your job is to create a binary classifier using TensorFlow to determine if a passenger survived or not. The `Survived` column lets you know if the person survived. Then, upload your predictions to Kaggle and submit your accuracy score at the end of this Colab, along with a brief conclusion.

To get the dataset, you'll need to accept the competition's rules by clicking the "I understand and accept" button on the competition rules page. Then upload your `kaggle.json` file and run the code below.

[ ]:
```
! chmod 600 kaggle.json && (ls ~/.kaggle 2>/dev/null || mkdir ~/.kaggle) && cp␣
↪kaggle.json ~/.kaggle/ && echo 'Done'
! kaggle competitions download -c titanic
! ls
```

```
kaggle.json
Done
titanic.zip: Skipping, found more recently modified local copy (use --force to
force download)
colab-key.zip  kaggle.json    slides.pptx    titanic.zip
colab.ipynb    slides.md      submission.csv
```

Note: If you see a "403 - Forbidden" error above, you still need to click "I understand and accept" on the competition rules page.

Three files are downloaded:

1. `train.csv`: training data (contains features and targets)
2. `test.csv`: feature data used to make predictions to send to Kaggle
3. `gender_submission.csv`: an example competition submission file

1.3 Step 1: Exploratory Data Analysis

Perform exploratory data analysis and data preprocessing. Use as many text and code blocks as you need to explore the data. Note any findings. Repair any data issues you find.

Student Solution

Imports and acquring data

[ ]:
```
import pandas as pd
import tensorflow as tf
import numpy as num
from zipfile import ZipFile
zip_file = ZipFile('titanic.zip')

train_df = pd.read_csv(zip_file.open('train.csv'))
test_df = pd.read_csv(zip_file.open('test.csv'))
# gender_df = pd.read_csv(zip_file.open('gender_submission.csv'))
```

[ ]:
```
test_df.sample(5)
```

[ ]:
```
     PassengerId  Pclass                                          Name     Sex  \
270         1162       1                  McCaffry, Mr. Thomas Francis    male
212         1104       2                     Deacon, Mr. Percy William    male
143         1035       2                    Beauchamp, Mr. Henry James    male
242         1134       1                  Spedden, Mr. Frederic Oakley    male
239         1131       1  Douglas, Mrs. Walter Donald (Mahala Dutton)  female

      Age  SibSp  Parch        Ticket       Fare Cabin Embarked
```

```
270  46.0      0      0         13050   75.2417   C6        C
212  17.0      0      0  S.O.C. 14879   73.5000   NaN       S
143  28.0      0      0        244358   26.0000   NaN       S
242  45.0      1      1         16966  134.5000   E34       C
239  48.0      1      0      PC 17761  106.4250   C86       C
```

The Challenge

In this challenge, we ask you to build a predictive model that answers the question: "what sorts of people were more likely to survive?" using passenger data (ie name, age, gender, socio-economic class, etc).

[ ]:
```
train_df.sample(10)
```

[ ]:
```
     PassengerId  Survived  Pclass                          Name     Sex  \
586          587         0       2           Jarvis, Mr. John Denzil    male
607          608         1       1         Daniel, Mr. Robert Williams    male
550          551         1       1         Thayer, Mr. John Borland Jr    male
497          498         0       3   Shellard, Mr. Frederick William    male
674          675         0       2        Watson, Mr. Ennis Hastings    male
178          179         0       2               Hale, Mr. Reginald    male
288          289         1       2             Hosono, Mr. Masabumi    male
60            61         0       3            Sirayanian, Mr. Orsen    male
218          219         1       1            Bazzani, Miss. Albina  female
744          745         1       3               Stranden, Mr. Juho    male

      Age  SibSp  Parch             Ticket      Fare Cabin Embarked
586  47.0      0      0             237565   15.0000   NaN        S
607  27.0      0      0             113804   30.5000   NaN        S
550  17.0      0      2              17421  110.8833   C70        C
497   NaN      0      0          C.A. 6212   15.1000   NaN        S
674   NaN      0      0             239856    0.0000   NaN        S
178  30.0      0      0             250653   13.0000   NaN        S
288  42.0      0      0             237798   13.0000   NaN        S
60   22.0      0      0               2669    7.2292   NaN        C
218  32.0      0      0              11813   76.2917   D15        C
744  31.0      0      0  STON/O 2. 3101288    7.9250   NaN        S
```

Preprocessing: Simplyfying the dataset

[ ]:
```
col = ['Survived', 'Pclass', 'Sex', 'Age', 'SibSp',
       'Parch','Fare', 'Embarked']

# train_df[col+['PassengerId']]
train_df[col]
```

```
[ ]:      Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
     0           0       3    male  22.0      1      0   7.2500        S
     1           1       1  female  38.0      1      0  71.2833        C
     2           1       3  female  26.0      0      0   7.9250        S
     3           1       1  female  35.0      1      0  53.1000        S
     4           0       3    male  35.0      0      0   8.0500        S
     ..        …       …      …     …    …      …      …        …
     886         0       2    male  27.0      0      0  13.0000        S
     887         1       1  female  19.0      0      0  30.0000        S
     888         0       3  female   NaN      1      2  23.4500        S
     889         1       1    male  26.0      0      0  30.0000        C
     890         0       3    male  32.0      0      0   7.7500        Q

     [891 rows x 8 columns]
```

```
[ ]: col = ['Survived', 'Pclass', 'Sex', 'Age', 'SibSp',
            'Parch','Fare', 'Embarked']

     train_df = train_df[col]
     #test_df  = test_df[col+['PassengerId']]

     #177 data points removed
     train_df.dropna(inplace = True)
     test_df.dropna(inplace=True)
     # print(train_df.sample(50))

     EmbarkedMap = {
            "S": 0,
            "C": 1,
            "Q": 2
     }
     # print(train_df["Embarked"])
     Embarked = train_df["Embarked"]
     NumericEmbarked = [EmbarkedMap[v] for v in Embarked]
     NumericSex = [0 if x == "male" else 1 for x in train_df["Sex"]]
     train_df["NumericEmbarked"] = NumericEmbarked
     train_df["NumericSex"] = NumericSex
     train_df
     # train_df = pd.get_dummies(train_df)
     #train_df





     # Embarked = test_df["Embarked"]
     # NumericEmbarked = [EmbarkedMap[v] for v in Embarked]
     # NumericSex = [0 if x == "male" else 1 for x in test_df["Sex"]]
```

```
# test_df["NumericEmbarked"] = NumericEmbarked
# test_df["NumericSex"] = NumericSex
# tes_df
```

[ ]:

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked  \
0           0       3    male  22.0      1      0   7.2500        S
1           1       1  female  38.0      1      0  71.2833        C
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
..         …      …      …    …    …     …       …      …
885         0       3  female  39.0      0      5  29.1250        Q
886         0       2    male  27.0      0      0  13.0000        S
887         1       1  female  19.0      0      0  30.0000        S
889         1       1    male  26.0      0      0  30.0000        C
890         0       3    male  32.0      0      0   7.7500        Q

     NumericEmbarked  NumericSex
0                  0           0
1                  1           1
2                  0           1
3                  0           1
4                  0           0
..                …          …
885                2           1
886                0           0
887                0           1
889                1           0
890                2           0

[712 rows x 10 columns]
```

[ ]:

```

```

[ ]:

```
# there are 177 rows missing ages!
print(og_df.isna().sum(),'\n')
# there are 86 rows missing ages!
print(og_df.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

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
```

[ ]:
```
target_column = 'Survived'
feature_columns = [c for c in train_df.columns if c != target_column and c!=␣
↪'Embarked' and c!= 'Sex']
target_column, feature_columns
```

[ ]:
```
('Survived',
 ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'NumericEmbarked', 'NumericSex'])
```

Using Standard Scalar

[ ]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# feature_columns
#train_df
train_df[feature_columns]
train_df[feature_columns] = scaler.fit_transform(train_df[feature_columns])
X_train = train_df[feature_columns]
y_train = train_df[target_column]
```

Getting target and feature columns

1.4 Step 2: The Model

Build, fit, and evaluate a classification model. Perform any model-specific data processing that you need to perform. If the toolkit you use supports it, create visualizations for loss and accuracy improvements. Use as many text and code blocks as you need to explore the data. Note any findings.

Student Solution

```
[ ]: model = tf.keras.Sequential([
         tf.keras.layers.Dense(64, activation=tf.nn.relu,
                               input_shape=(X_train.columns.size,)),
         tf.keras.layers.Dense(32, activation=tf.nn.relu),
         tf.keras.layers.Dense(16, activation=tf.nn.relu),
         tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
     ])
```

```
[ ]: model.compile(
         loss='binary_crossentropy',
         optimizer='Adam',
         metrics=['accuracy']
     )
```

Early Stopping

```
[ ]: es = tf.keras.callbacks.EarlyStopping(
         monitor='loss',
         min_delta=1e-3,
         patience=5
     )

     complete_model = model.fit(
         X_train,
         y_train,
         epochs=500,
         verbose=2,
         callbacks=[es]
     )
     print(complete_model.history['accuracy'][-1])
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: <class 'pandas.core.frame.DataFrame'>,
<class 'NoneType'>
Train on 712 samples
Epoch 1/500
712/712 - 0s - loss: 0.6501 - accuracy: 0.6756
Epoch 2/500
712/712 - 0s - loss: 0.5472 - accuracy: 0.7893
Epoch 3/500
712/712 - 0s - loss: 0.4770 - accuracy: 0.7879
Epoch 4/500
712/712 - 0s - loss: 0.4475 - accuracy: 0.7907
Epoch 5/500
712/712 - 0s - loss: 0.4314 - accuracy: 0.8034
Epoch 6/500
712/712 - 0s - loss: 0.4217 - accuracy: 0.8202
Epoch 7/500
```

```
712/712 - 0s - loss: 0.4159 - accuracy: 0.8230
Epoch 8/500
712/712 - 0s - loss: 0.4108 - accuracy: 0.8301
Epoch 9/500
712/712 - 0s - loss: 0.4050 - accuracy: 0.8301
Epoch 10/500
712/712 - 0s - loss: 0.4034 - accuracy: 0.8329
Epoch 11/500
712/712 - 0s - loss: 0.3985 - accuracy: 0.8329
Epoch 12/500
712/712 - 0s - loss: 0.3950 - accuracy: 0.8287
Epoch 13/500
712/712 - 0s - loss: 0.3923 - accuracy: 0.8399
Epoch 14/500
712/712 - 0s - loss: 0.3901 - accuracy: 0.8427
Epoch 15/500
712/712 - 0s - loss: 0.3880 - accuracy: 0.8441
Epoch 16/500
712/712 - 0s - loss: 0.3855 - accuracy: 0.8427
Epoch 17/500
712/712 - 0s - loss: 0.3844 - accuracy: 0.8469
Epoch 18/500
712/712 - 0s - loss: 0.3832 - accuracy: 0.8497
Epoch 19/500
712/712 - 0s - loss: 0.3793 - accuracy: 0.8455
Epoch 20/500
712/712 - 0s - loss: 0.3791 - accuracy: 0.8483
Epoch 21/500
712/712 - 0s - loss: 0.3750 - accuracy: 0.8455
Epoch 22/500
712/712 - 0s - loss: 0.3747 - accuracy: 0.8469
Epoch 23/500
712/712 - 0s - loss: 0.3717 - accuracy: 0.8525
Epoch 24/500
712/712 - 0s - loss: 0.3716 - accuracy: 0.8511
Epoch 25/500
712/712 - 0s - loss: 0.3691 - accuracy: 0.8553
Epoch 26/500
712/712 - 0s - loss: 0.3675 - accuracy: 0.8525
Epoch 27/500
712/712 - 0s - loss: 0.3670 - accuracy: 0.8525
Epoch 28/500
712/712 - 0s - loss: 0.3666 - accuracy: 0.8469
Epoch 29/500
712/712 - 0s - loss: 0.3614 - accuracy: 0.8553
Epoch 30/500
712/712 - 0s - loss: 0.3625 - accuracy: 0.8539
Epoch 31/500
```

```
712/712 - 0s - loss: 0.3615 - accuracy: 0.8525
Epoch 32/500
712/712 - 0s - loss: 0.3631 - accuracy: 0.8624
Epoch 33/500
712/712 - 0s - loss: 0.3606 - accuracy: 0.8511
Epoch 34/500
712/712 - 0s - loss: 0.3606 - accuracy: 0.8511
0.8511236
```

[ ]:

```

```

[ ]:

```
test_df = pd.read_csv(zip_file.open('test.csv'))

col = ['Survived', 'Pclass', 'Sex', 'Age', 'SibSp',
       'Parch','Fare', 'Embarked', ]

test = test_df.filter(items=col).copy()


#177 data points removed
# test.dropna(inplace = True)
# print(train_df.sample(50))

EmbarkedMap = {
       "S": 0,
       "C": 1,
       "Q": 2
}
# print(train_df["Embarked"])
Embarked = test["Embarked"]
NumericEmbarked = [EmbarkedMap[v] for v in Embarked]
NumericSex = [0 if x == "male" else 1 for x in test["Sex"]]
test["NumericEmbarked"] = NumericEmbarked
test["NumericSex"] = NumericSex
test
# train_df = pd.get_dummies(train_df)
#train_df
```

[ ]:

```
     Pclass     Sex   Age  SibSp  Parch      Fare Embarked  NumericEmbarked  \
0         3    male  34.5      0      0    7.8292        Q                2
1         3  female  47.0      1      0    7.0000        S                0
2         2    male  62.0      0      0    9.6875        Q                2
3         3    male  27.0      0      0    8.6625        S                0
4         3  female  22.0      1      1   12.2875        S                0
..      …       …   …      …      …        …         …
413       3    male   NaN      0      0    8.0500        S                0
414       1  female  39.0      0      0  108.9000        C                1
```

```
415       3    male  38.5      0      0     7.2500        S                 0
416       3    male   NaN      0      0     8.0500        S                 0
417       3    male   NaN      1      1    22.3583        C                 1

     NumericSex
0             0
1             1
2             0
3             0
4             1
..          …
413           0
414           1
415           0
416           0
417           0

[418 rows x 9 columns]
```

[ ]:

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# feature_columns
#train_df
test[feature_columns]
test[feature_columns] = scaler.fit_transform(test[feature_columns])
test = test[feature_columns].copy()
#y_train = test[target_column]
```

[ ]:

```python
print(len(test))
```

```
87
```

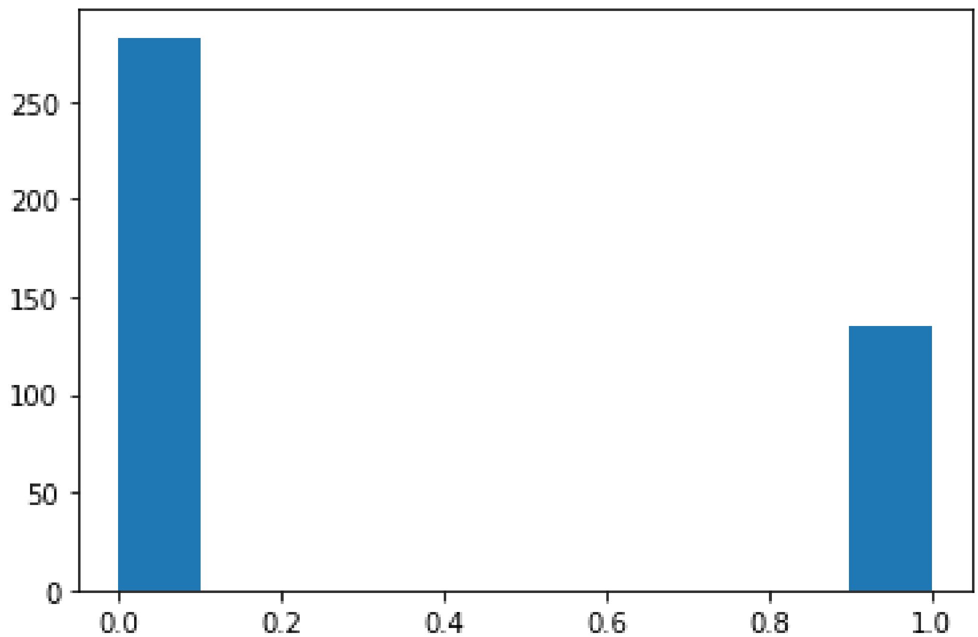
[ ]:

```python
import matplotlib.pyplot as plt


predictions = model.predict(test[feature_columns])

def newRound(i):
    if i>.5:
        return 1
    else:
        return 0
predictions = [newRound(x) for x in predictions]
plt.hist(predictions)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
```

```
data adapter that can handle input: <class 'pandas.core.frame.DataFrame'>,
<class 'NoneType'>
```

[ ]:
```
(array([283.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 135.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



[ ]:
```

```

1.5 Step 3: Make Predictions and Upload To Kaggle

In this step you will make predictions on the features found in the `test.csv` file and upload them to Kaggle using the Kaggle API. Use as many text and code blocks as you need to explore the data. Note any findings.

Student Solution

[ ]:
```

```

[ ]:
```
# test_df['PassengerId']
# predictions
submit_df = pd.DataFrame({
    "PassengerId": test_df['PassengerId'],
    "Survived": predictions
})

submit_df.to_csv(path_or_buf="./submission.csv", index=False)
```

[ ]:
```
! cat submission.csv | wc -l
```

```
419
```

[ ]:
```
# test_df.to_csv(path_or_buf= './submission.csv', index=False)
```

[ ]:
```
!kaggle competitions submit -c titanic -f submission.csv -m "Message"
```

```
100%|                    | 2.77k/2.77k [00:00<00:00, 3.46kB/s]
Successfully submitted to Titanic - Machine Learning from Disaster
```

[ ]:
```
!kaggle competitions submissions titanic
```

```
fileName        date                 description  status    publicScore
privateScore
--------------  -------------------  -----------  --------  -----------
------------
submission.csv  2021-06-30 19:52:56  Message      complete  0.71531      None
submission.csv  2021-06-30 19:51:20  Message      complete  0.74401      None
submission.csv  2021-06-30 19:50:06  Message      complete  0.75119      None
submission.csv  2021-06-30 19:48:01  Message      complete  0.76794      None
submission.csv  2021-06-30 19:37:47  Message      complete  0.74880      None
submission.csv  2021-06-30 19:35:33  Message      error     None         None
submission.csv  2021-06-30 19:33:30  Message      error     None         None
submission.csv  2021-06-30 18:18:30  None         error     None         None
```

[ ]:
```

```

[ ]:
```
! head submission.csv
```

```
PassengerId,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
892,3,"Kelly, Mr. James",male,34.5,0,0,330911,7.8292,,Q
893,3,"Wilkes, Mrs. James (Ellen Needs)",female,47.0,1,0,363272,7.0,,S
894,2,"Myles, Mr. Thomas Francis",male,62.0,0,0,240276,9.6875,,Q
895,3,"Wirz, Mr. Albert",male,27.0,0,0,315154,8.6625,,S
896,3,"Hirvonen, Mrs. Alexander (Helga E
Lindqvist)",female,22.0,1,1,3101298,12.2875,,S
897,3,"Svensson, Mr. Johan Cervin",male,14.0,0,0,7538,9.225,,S
898,3,"Connolly, Miss. Kate",female,30.0,0,0,330972,7.6292,,Q
899,2,"Caldwell, Mr. Albert Francis",male,26.0,1,1,248738,29.0,,S
900,3,"Abrahim, Mrs. Joseph (Sophie Halaut Easu)",female,18.0,0,0,2657,7.2292,,C
```

What was your Kaggle score?

75

1.6 Step 4: Iterate on Your Model

In this step you're encouraged to play around with your model settings and to even try different models. See if you can get a better score. Use as many text and code blocks as you need to explore the data. Note any findings.

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation=tf.nn.leaky_relu,
                          input_shape=(X_train.columns.size,)),
    tf.keras.layers.Dense(32, activation=tf.nn.leaky_relu),
    tf.keras.layers.Dense(16, activation=tf.nn.leaky_relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
```

[ ]:
```
model.compile(
    loss='binary_crossentropy',
    optimizer='Adam',
    metrics=['accuracy']
)
```

[ ]:
```
es = tf.keras.callbacks.EarlyStopping(
    monitor='loss',
    min_delta=1e-3,
    patience=5
)

complete_model = model.fit(
    X_train,
    y_train,
    epochs=500,
    verbose=2,
    callbacks=[es]
)

print(complete_model.history['accuracy'][-1])
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: <class 'pandas.core.frame.DataFrame'>,
<class 'NoneType'>
Train on 712 samples
Epoch 1/500
712/712 - 0s - loss: 0.6352 - accuracy: 0.7008
Epoch 2/500
712/712 - 0s - loss: 0.5252 - accuracy: 0.8034
Epoch 3/500
712/712 - 0s - loss: 0.4687 - accuracy: 0.7992
Epoch 4/500
712/712 - 0s - loss: 0.4399 - accuracy: 0.8118
Epoch 5/500
```

```
712/712 - 0s - loss: 0.4272 - accuracy: 0.8160
Epoch 6/500
712/712 - 0s - loss: 0.4196 - accuracy: 0.8244
Epoch 7/500
712/712 - 0s - loss: 0.4142 - accuracy: 0.8301
Epoch 8/500
712/712 - 0s - loss: 0.4118 - accuracy: 0.8315
Epoch 9/500
712/712 - 0s - loss: 0.4054 - accuracy: 0.8287
Epoch 10/500
712/712 - 0s - loss: 0.4047 - accuracy: 0.8329
Epoch 11/500
712/712 - 0s - loss: 0.4011 - accuracy: 0.8272
Epoch 12/500
712/712 - 0s - loss: 0.3990 - accuracy: 0.8287
Epoch 13/500
712/712 - 0s - loss: 0.3981 - accuracy: 0.8357
Epoch 14/500
712/712 - 0s - loss: 0.3976 - accuracy: 0.8441
Epoch 15/500
712/712 - 0s - loss: 0.3929 - accuracy: 0.8343
Epoch 16/500
712/712 - 0s - loss: 0.3912 - accuracy: 0.8343
Epoch 17/500
712/712 - 0s - loss: 0.3897 - accuracy: 0.8371
Epoch 18/500
712/712 - 0s - loss: 0.3887 - accuracy: 0.8399
Epoch 19/500
712/712 - 0s - loss: 0.3847 - accuracy: 0.8427
Epoch 20/500
712/712 - 0s - loss: 0.3837 - accuracy: 0.8385
Epoch 21/500
712/712 - 0s - loss: 0.3826 - accuracy: 0.8371
Epoch 22/500
712/712 - 0s - loss: 0.3804 - accuracy: 0.8455
Epoch 23/500
712/712 - 0s - loss: 0.3819 - accuracy: 0.8399
Epoch 24/500
712/712 - 0s - loss: 0.3789 - accuracy: 0.8399
Epoch 25/500
712/712 - 0s - loss: 0.3776 - accuracy: 0.8413
Epoch 26/500
712/712 - 0s - loss: 0.3759 - accuracy: 0.8469
Epoch 27/500
712/712 - 0s - loss: 0.3753 - accuracy: 0.8455
Epoch 28/500
712/712 - 0s - loss: 0.3772 - accuracy: 0.8413
Epoch 29/500
```

```
712/712 - 0s - loss: 0.3748 - accuracy: 0.8497
Epoch 30/500
712/712 - 0s - loss: 0.3720 - accuracy: 0.8469
Epoch 31/500
712/712 - 0s - loss: 0.3724 - accuracy: 0.8427
Epoch 32/500
712/712 - 0s - loss: 0.3726 - accuracy: 0.8413
Epoch 33/500
712/712 - 0s - loss: 0.3661 - accuracy: 0.8539
Epoch 34/500
712/712 - 0s - loss: 0.3671 - accuracy: 0.8483
Epoch 35/500
712/712 - 0s - loss: 0.3654 - accuracy: 0.8469
Epoch 36/500
712/712 - 0s - loss: 0.3641 - accuracy: 0.8427
Epoch 37/500
712/712 - 0s - loss: 0.3624 - accuracy: 0.8525
Epoch 38/500
712/712 - 0s - loss: 0.3627 - accuracy: 0.8455
Epoch 39/500
712/712 - 0s - loss: 0.3623 - accuracy: 0.8413
Epoch 40/500
712/712 - 0s - loss: 0.3620 - accuracy: 0.8497
Epoch 41/500
712/712 - 0s - loss: 0.3604 - accuracy: 0.8539
Epoch 42/500
712/712 - 0s - loss: 0.3567 - accuracy: 0.8539
Epoch 43/500
712/712 - 0s - loss: 0.3576 - accuracy: 0.8511
Epoch 44/500
712/712 - 0s - loss: 0.3550 - accuracy: 0.8525
Epoch 45/500
712/712 - 0s - loss: 0.3543 - accuracy: 0.8525
Epoch 46/500
712/712 - 0s - loss: 0.3513 - accuracy: 0.8553
Epoch 47/500
712/712 - 0s - loss: 0.3498 - accuracy: 0.8539
Epoch 48/500
712/712 - 0s - loss: 0.3526 - accuracy: 0.8525
Epoch 49/500
712/712 - 0s - loss: 0.3530 - accuracy: 0.8497
Epoch 50/500
712/712 - 0s - loss: 0.3481 - accuracy: 0.8567
Epoch 51/500
712/712 - 0s - loss: 0.3488 - accuracy: 0.8469
Epoch 52/500
712/712 - 0s - loss: 0.3572 - accuracy: 0.8581
Epoch 53/500
```

```
712/712 - 0s - loss: 0.3614 - accuracy: 0.8413
Epoch 54/500
712/712 - 0s - loss: 0.3526 - accuracy: 0.8441
Epoch 55/500
712/712 - 0s - loss: 0.3434 - accuracy: 0.8553
Epoch 56/500
712/712 - 0s - loss: 0.3439 - accuracy: 0.8596
Epoch 57/500
712/712 - 0s - loss: 0.3439 - accuracy: 0.8497
Epoch 58/500
712/712 - 0s - loss: 0.3451 - accuracy: 0.8511
Epoch 59/500
712/712 - 0s - loss: 0.3405 - accuracy: 0.8581
Epoch 60/500
712/712 - 0s - loss: 0.3453 - accuracy: 0.8511
Epoch 61/500
712/712 - 0s - loss: 0.3397 - accuracy: 0.8525
Epoch 62/500
712/712 - 0s - loss: 0.3406 - accuracy: 0.8666
Epoch 63/500
712/712 - 0s - loss: 0.3409 - accuracy: 0.8638
Epoch 64/500
712/712 - 0s - loss: 0.3369 - accuracy: 0.8624
Epoch 65/500
712/712 - 0s - loss: 0.3355 - accuracy: 0.8638
Epoch 66/500
712/712 - 0s - loss: 0.3370 - accuracy: 0.8610
Epoch 67/500
712/712 - 0s - loss: 0.3351 - accuracy: 0.8581
Epoch 68/500
712/712 - 0s - loss: 0.3357 - accuracy: 0.8567
Epoch 69/500
712/712 - 0s - loss: 0.3359 - accuracy: 0.8483
Epoch 70/500
712/712 - 0s - loss: 0.3462 - accuracy: 0.8525
0.8525281
```

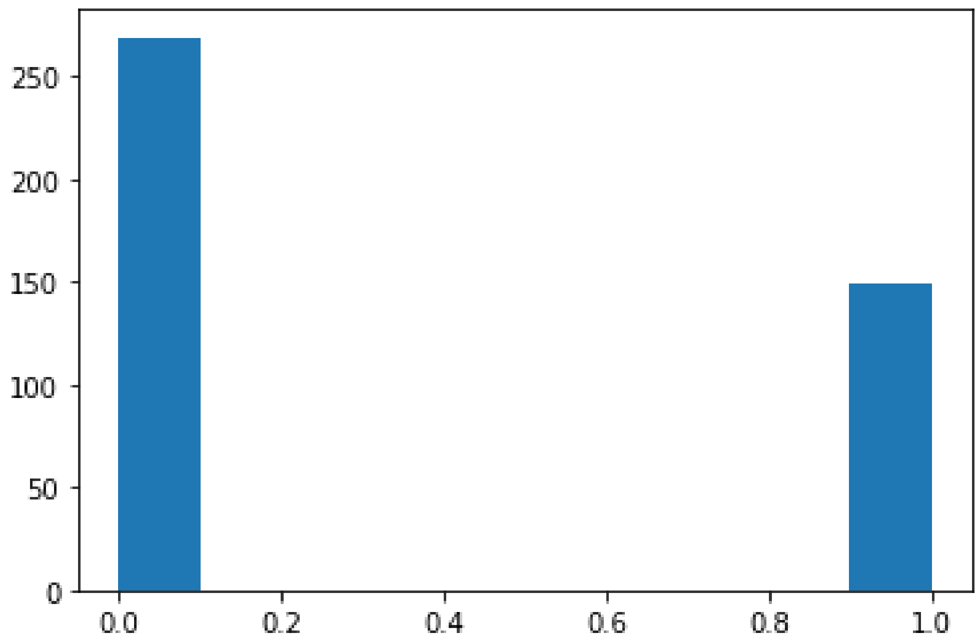
[ ]:

```
predictions = model.predict(test[feature_columns])

def newRound(i):
    if i>.5:
        return 1
    else:
        return 0
predictions = [newRound(x) for x in predictions]
plt.hist(predictions)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
```

```
data adapter that can handle input: <class 'pandas.core.frame.DataFrame'>,
<class 'NoneType'>
```

[ ]:
```
(array([269.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 149.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



[ ]:
```
submit_df = pd.DataFrame({
    "PassengerId": test_df['PassengerId'],
    "Survived": predictions
})

submit_df.to_csv(path_or_buf="./submission.csv", index=False)
```

1.6.1 Conclusion

The original model implemented return a kaggle score of 75. After iterating through hyperparameters for the model used, changing the activation function to `leaky model`. This returned a kaggle score of 77.
