Titanics Dataset Predictions

Jose M. Serra Jr.

Mounted Google Drive to import data from Kaggle Train, and Test data sets.

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [2]:
```
import matplotlib.pyplot as mplplt
import numpy as np
import pandas as pd
from scipy.stats import reciprocal
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.callbacks import *
from tensorflow.keras.layers import *
from tensorflow.keras.metrics import *
from tensorflow.keras.models import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.optimizers.schedules import *
import tensorflow.keras.callbacks as tkc
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.compose import make_column_transformer, ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.manifold import Isomap
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder, LabelEncoder
from sklearn.utils import resample
```

Here I included standard packages which include TensorFlow machine learning library, as well as the Sci-Kit Learn library for data processing. As well as standard imports such as Pandas for reading data, and numpy for numerial analysis.

In [3]:
```
file1_train, file2_test = pd.read_csv(r"/content/drive/MyDrive/Titanic/train.csv", delimiter=","),\
pd.read_csv(r"/content/drive/MyDrive/Titanic/test.csv", delimiter= ",")
```

Now I am going to encode the Sex column into a binary 1s, and 0s output Sex_Binary column.

In [4]:
```
file1_train = file1_train.drop(columns = ["PassengerId","Name"])
X, y = file1_train.drop("Survived", axis=1), file1_train["Survived"]
```

In [6]:
```
file1_train.head()
```

Out[6]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	1	female	35.0	1	0	113803	53.1000	C123	S
4	0	3	male	35.0	0	0	373450	8.0500	NaN	S

In [7]:
```
objs = X.select_dtypes(["object"])
num  = X.select_dtypes(["number"])
```

In [8]:
```
objs.isnull().sum().head()
```

Out[8]:
```
Sex           0
Ticket        0
Cabin       687
Embarked      2
dtype: int64
```

In [9]:
```
num.isnull().sum().head()
```

Out[9]:
```
Pclass      0
Age       177
```

```
SibSp       0
Parch       0
Fare        0
dtype: int64
```

In [10]:
```
numerical_features = num.columns
numerical_pipeline = Pipeline(
    steps=[
            ("imputer", SimpleImputer(strategy = 'mean')),
            ("scaler", StandardScaler())
])
```

In [11]:
```
for i in range(len(objs.columns)):
    print(objs.columns[i], objs.iloc[:,i].value_counts().unique())
```

```
Sex [577 314]
Ticket [7 6 5 4 3 2 1]
Cabin [4 3 2 1]
Embarked [644 168  77]
```

In [12]:
```
binary = ["Sex"]
binary_pipeline = Pipeline(steps=[("binary", OneHotEncoder())])
```

In [13]:
```
cat1 = ["Ticket", "Cabin", "Embarked"]
catergorical_pipeline = Pipeline(steps=[("imputer", SimpleImputer(strategy = 'most_frequent')), ("ordinal_en
```

In [14]:
```
data_preprocessor = ColumnTransformer( [('numerical', numerical_pipeline, numerical_features),
                                        ('binary', binary_pipeline, binary),
                                        ('categorical', catergorical_pipeline, cat1)])
```

In [15]:
```
X.head()
```

Out[15]:

	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	female	38.0	1	0	PC 17599	71.2833	C85	C

	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	female	35.0	1	0	113803	53.1000	C123	S

In [16]:
```
X = Pipeline(steps=[('processing',data_preprocessor)]).fit_transform(X)
```

In [17]:
```
param_distribs = {'learn_rate' : np.array(np.linspace(.1,.9))}
```

In [18]:
```
def base_model1(learn_rate = .1):
    input_dim = X.shape[1]
    model =Sequential([
    Dense(200 , input_dim = input_dim, activation= "relu"),
    Dropout(rate=.10),
    Dense(100, activation= "tanh"),
    Dense(1,activation = "sigmoid"),
    ])
    lr_schedule = ExponentialDecay(
    learn_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)
    model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam(learning_rate=lr_schedule),
    return model
```

In [19]:
```
checkpoint = [ModelCheckpoint("Titanic.h5", monitor='accuracy', verbose=1, save_best_only=True, save_weights
early = EarlyStopping(monitor='accuracy', min_delta=0, patience=10, verbose=1, mode='auto')
```

In [21]:
```
NN_clf = KerasClassifier(build_fn=base_model1, epochs=100, verbose=1, callbacks =[checkpoint,early] )
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: KerasClassifier is depre
cated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  """Entry point for launching an IPython kernel.
```

In [23]:
```
%%capture
random_trainor = GridSearchCV(estimator=NN_clf,param_grid=param_distribs, cv=None)
random_trainor.fit(X,(y.values.reshape(-1,1)))
```

In [24]:
```
best = random_trainor.best_estimator_.model
```

In [25]:
```
best.summary()
```

```
Model: "sequential_413"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_1239 (Dense)           (None, 200)               2200

 dropout_413 (Dropout)        (None, 200)               0

 dense_1240 (Dense)           (None, 100)               20100

 dense_1241 (Dense)           (None, 1)                 101

=================================================================
Total params: 22,401
Trainable params: 22,401
Non-trainable params: 0
_________________________________________________________________
```

In [26]:
```
best.save("Titanic.h5")
```

In [31]:
```
X_test = data_preprocessor.fit_transform(file2_test)
```

In [32]:
```
PassengerId = file2_test["PassengerId"].to_list()
```

In [33]:
```
final_pred = (best.predict(X_test) > 0.5).astype("int32").flatten()
```

In [30]:
```
#d = {"PassengerId":PassengerId, "Survived":final_pred}
#pd.DataFrame(data=d,index=None, columns= ["PassengerId","Survived"]).to_csv("12232021.csv",index=False, hea
```

In [ ]:

In [ ]: