Payments prediction with Neural Network

In this notebook we shall provide the prediction of default payments made by clients in Taiwan from April to Semptember 2005. The execution of the Neural Network will be made step by step.

Importing libraries

In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.model_selection import train_test_split
import joblib

%matplotlib inline
%config InlineBackend.figure_format='retina'

sns.set(style='whitegrid', palette='muted', font_scale=1.5)

rcParams['figure.figsize'] = 16,10

RANDOM_SEED = 60

np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)
```

In [2]:
```
X_test = pd.read_csv('data/X_test.csv')
X_train = pd.read_csv('data/X_train.csv')
y_train = pd.read_csv('data/y_train.csv')
```

Exploration

In [3]:
```
print(f"Shape x test {X_test.shape}")
print(f"Shape x train {X_train.shape}")
print(f"Shape y train {y_train.shape}")
```

```
Shape x test (6000, 24)
Shape x train (24000, 24)
Shape y train (24000, 2)
```

In [4]:
```
X_train.columns
```

Out[4]:
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'],
      dtype='object')
```

In [5]:
```
X_test.columns
```

Out[5]:
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
```

```
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'],
      dtype='object')
```

In [6]:
```
y_train.columns
```

Out[6]:
```
Index(['ID', 'default.payment.next.month'], dtype='object')
```

In [7]:
```
## First we verify if we have any missing data

missing = X_train.isnull().sum()
missing[missing > 0].sort_values(ascending=False)
```

Out[7]:
```
Series([], dtype: int64)
```

In [8]:
```
missing_y = y_train.isnull().sum()
missing_y[missing_y > 0].sort_values(ascending=False)
```

Out[8]:
```
Series([], dtype: int64)
```

In [9]:
```
X_train.index = X_train.ID
X_test.index = X_test.ID
```

In [10]:
```
# Droppping the ID column
X_train.drop('ID',axis=1,inplace=True)
X_test.drop('ID',axis=1,inplace=True)
```

In [11]:
```
X_train.head()
```

Out[11]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5
21754	80000.0	2	2	2	24	0	0	0	0	0
252	30000.0	1	2	2	28	0	0	0	0	0
22942	180000.0	2	5	1	44	0	0	-1	-1	-1
619	60000.0	1	1	2	25	0	0	0	0	0
17091	130000.0	2	2	2	25	0	0	0	0	0

5 rows × 23 columns

In [12]:
```
X_test.head()
```

Out[12]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5
2309	30000.0	1	2	2	25	0	0	0	0	0
22405	150000.0	2	1	2	26	0	0	0	0	0
23398	70000.0	2	3	1	32	0	0	0	0	0
25059	130000.0	1	3	2	49	0	0	0	0	0
2665	50000.0	2	2	2	36	0	0	0	0	0

5 rows × 23 columns

In [13]:
```
y_train = y_train.rename(columns={"default.payment.next.month":"def_payment"}
```

In [14]:
```
X_test.isnull().sum()
```

Out[14]:
```
LIMIT_BAL    0
SEX          0
EDUCATION    0
MARRIAGE     0
AGE          0
PAY_0        0
PAY_2        0
PAY_3        0
PAY_4        0
PAY_5        0
PAY_6        0
BILL_AMT1    0
BILL_AMT2    0
BILL_AMT3    0
BILL_AMT4    0
BILL_AMT5    0
BILL_AMT6    0
PAY_AMT1     0
PAY_AMT2     0
PAY_AMT3     0
PAY_AMT4     0
PAY_AMT5     0
PAY_AMT6     0
dtype: int64
```

In [15]:
```
X_train.SEX.value_counts(dropna=False)
```

Out[15]:
```
2    14518
1     9482
Name: SEX, dtype: int64
```

In [16]:
```
X_train.EDUCATION.value_counts(dropna=False)
```

Out[16]:
```
2    11186
1     8481
3     3959
5      224
4       97
6       43
0       10
Name: EDUCATION, dtype: int64
```

In [17]:
```
X_train = X_train.rename(columns={"PAY_0":"PAY_1"})
X_test = X_test.rename(columns={"PAY_0":"PAY_1"})

X_train.head()
```

Out[17]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
ID										
21754	80000.0	2	2	2	24	0	0	0	0	0
252	30000.0	1	2	2	28	0	0	0	0	0
22942	180000.0	2	5	1	44	0	0	-1	-1	-1

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
619	60000.0	1	1	2	25	0	0	0	0	0
17091	130000.0	2	2	2	25	0	0	0	0	0

5 rows × 23 columns

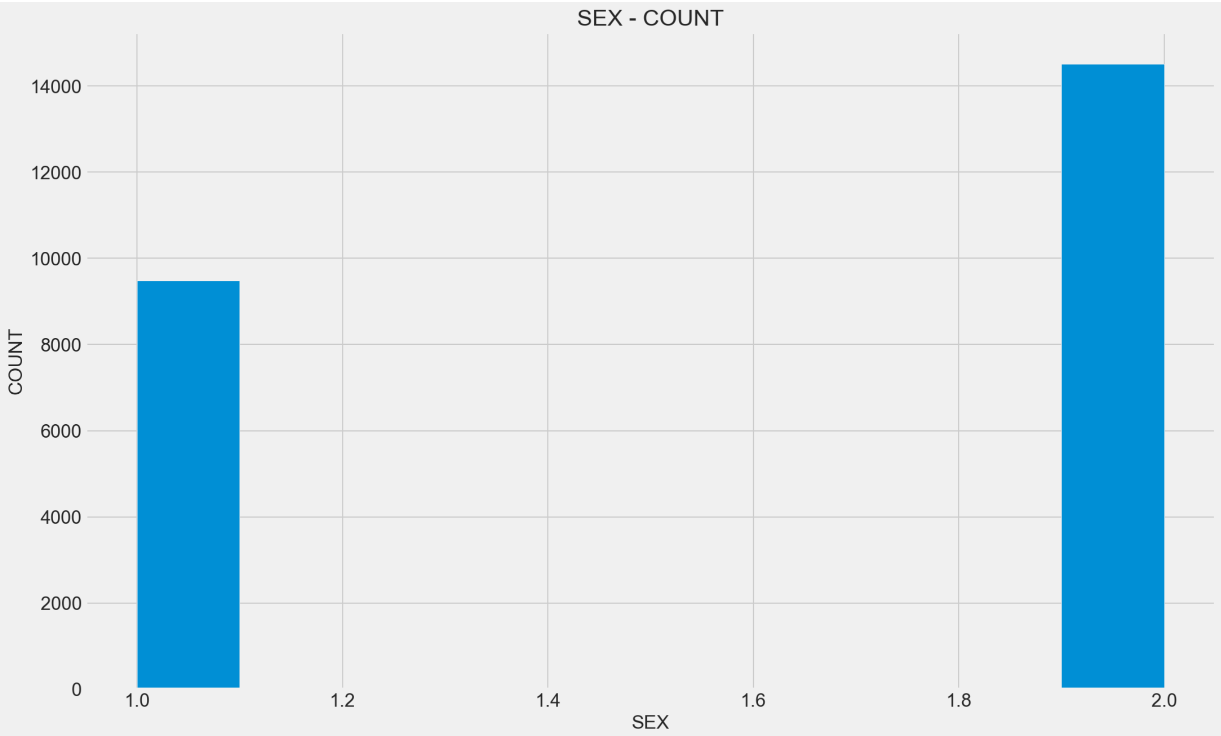
In [18]:
```
X_train.columns
```

Out[18]:
```
Index(['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_1', 'PAY_2',
       'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'],
      dtype='object')
```

Data visualization

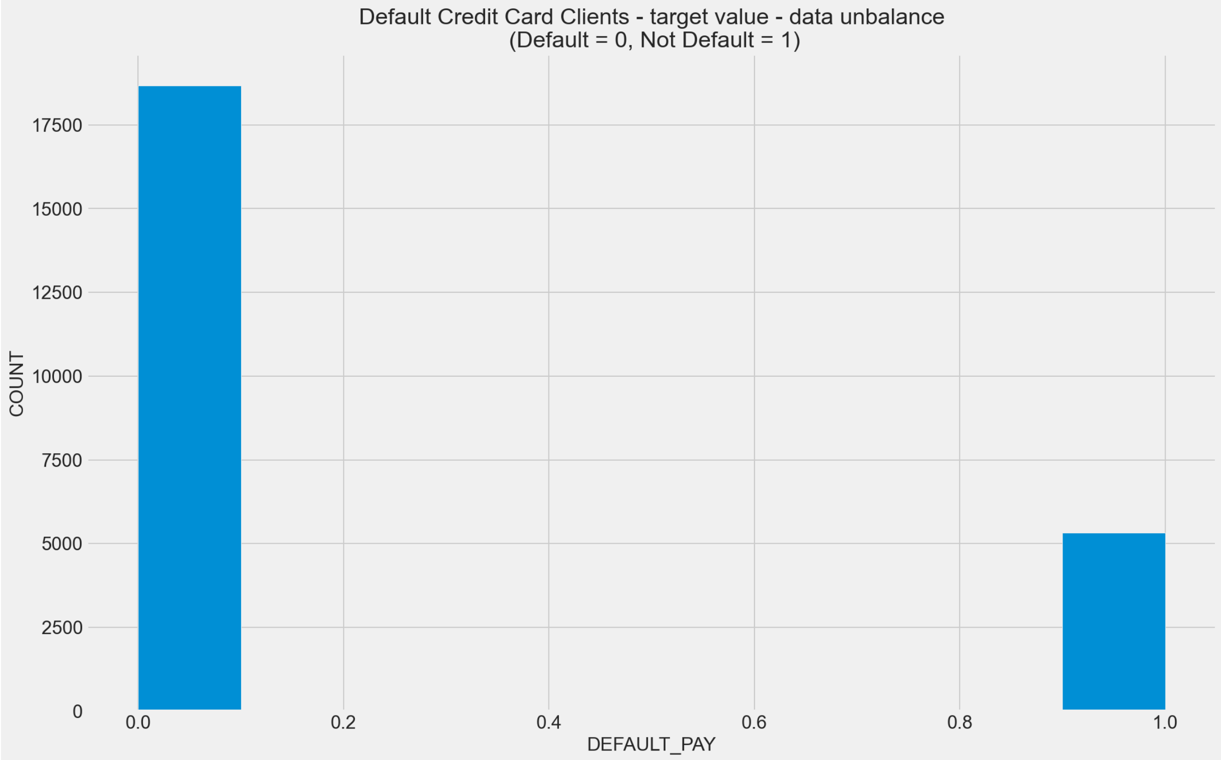
In [19]:
```
plt.style.use('fivethirtyeight')
X_train.SEX.hist()
plt.xlabel('SEX')
plt.ylabel('COUNT')
plt.title('SEX - COUNT')
```

Out[19]:
```
Text(0.5, 1.0, 'SEX - COUNT')
```



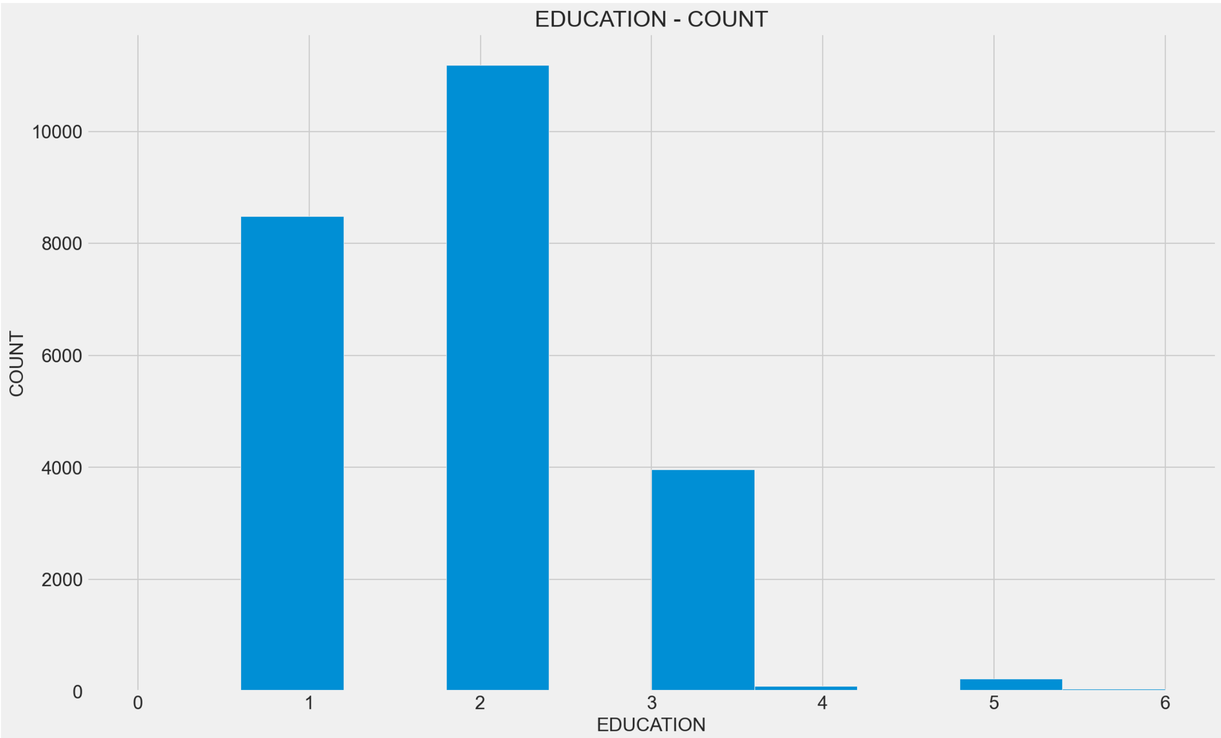
In [20]:
```
plt.style.use('fivethirtyeight')
y_train.def_payment.hist()
plt.xlabel('DEFAULT_PAY')
plt.ylabel('COUNT')
plt.title('Default Credit Card Clients - target value - data unbalance\n (Def
```

Out[20]:
```
Text(0.5, 1.0, 'Default Credit Card Clients - target value - data unbalance\n
(Default = 0, Not Default = 1)')
```



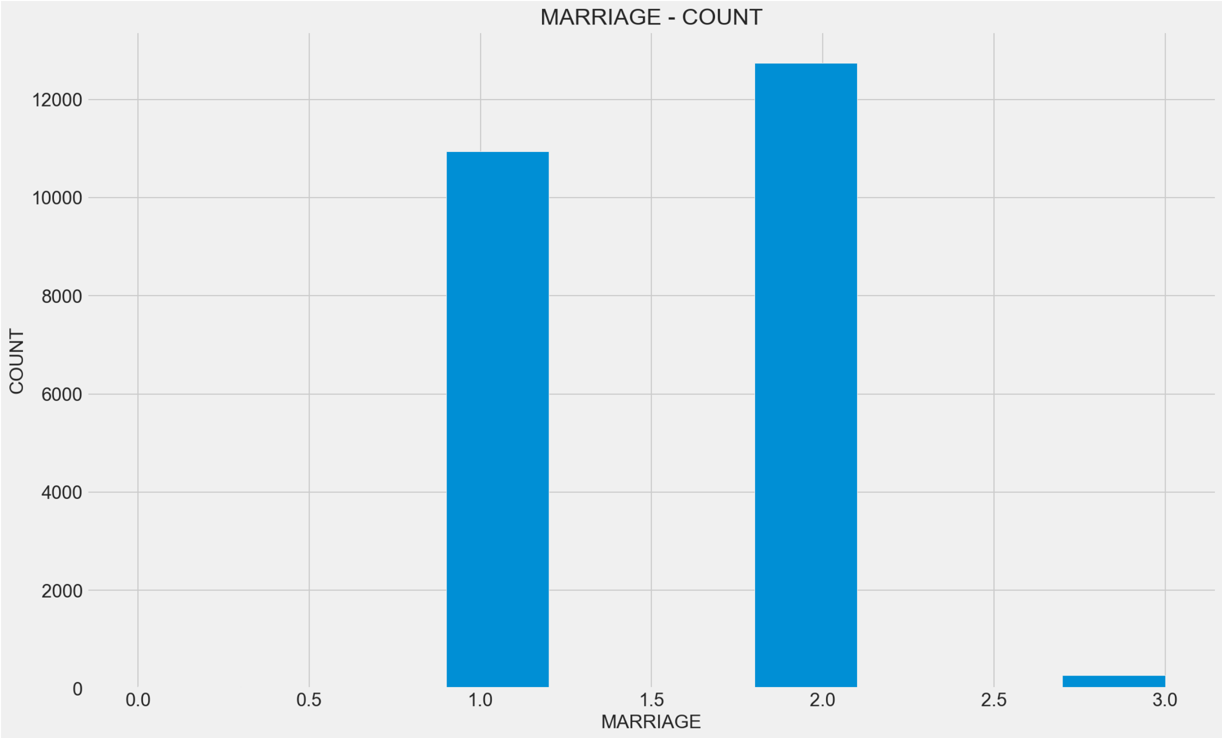
In [21]:
```
plt.style.use('fivethirtyeight')
X_train.EDUCATION.hist()
plt.xlabel('EDUCATION')
plt.ylabel('COUNT')
plt.title('EDUCATION - COUNT')
```

Out[21]:
```
Text(0.5, 1.0, 'EDUCATION - COUNT')
```



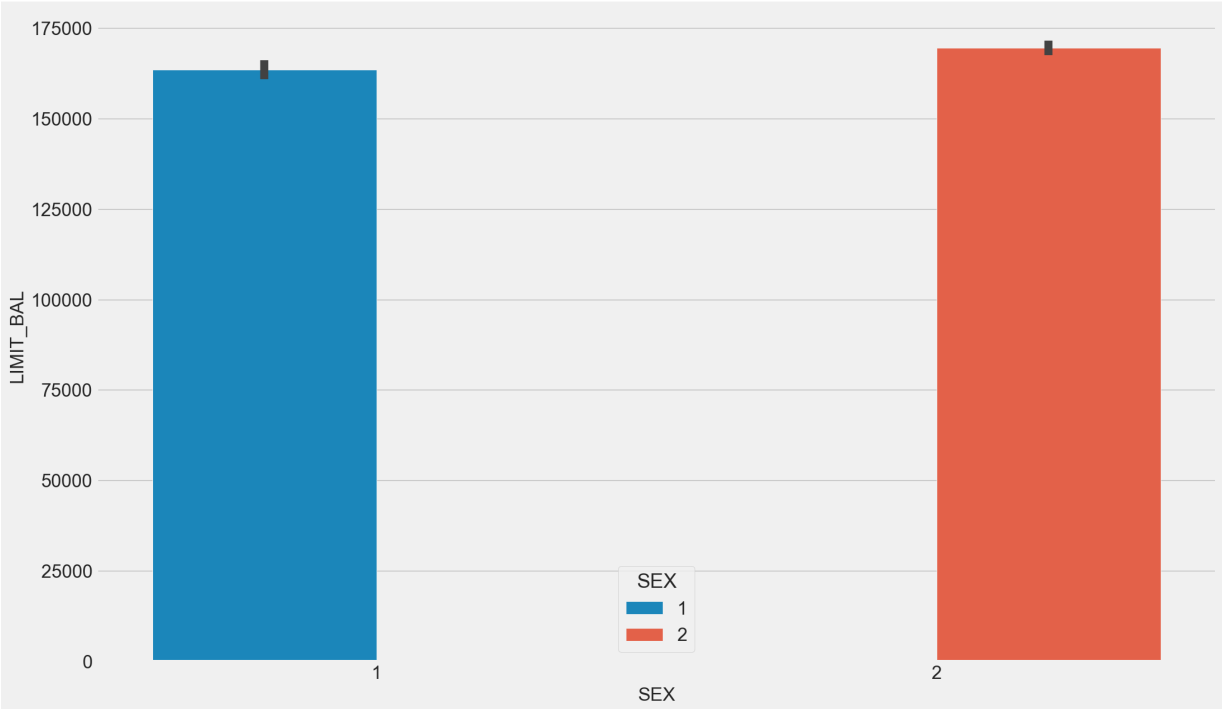
In [22]:
```
plt.style.use('fivethirtyeight')
X_train.MARRIAGE.hist()
plt.xlabel('MARRIAGE')
plt.ylabel('COUNT')
plt.title('MARRIAGE - COUNT')
```

Out[22]:
```
Text(0.5, 1.0, 'MARRIAGE - COUNT')
```



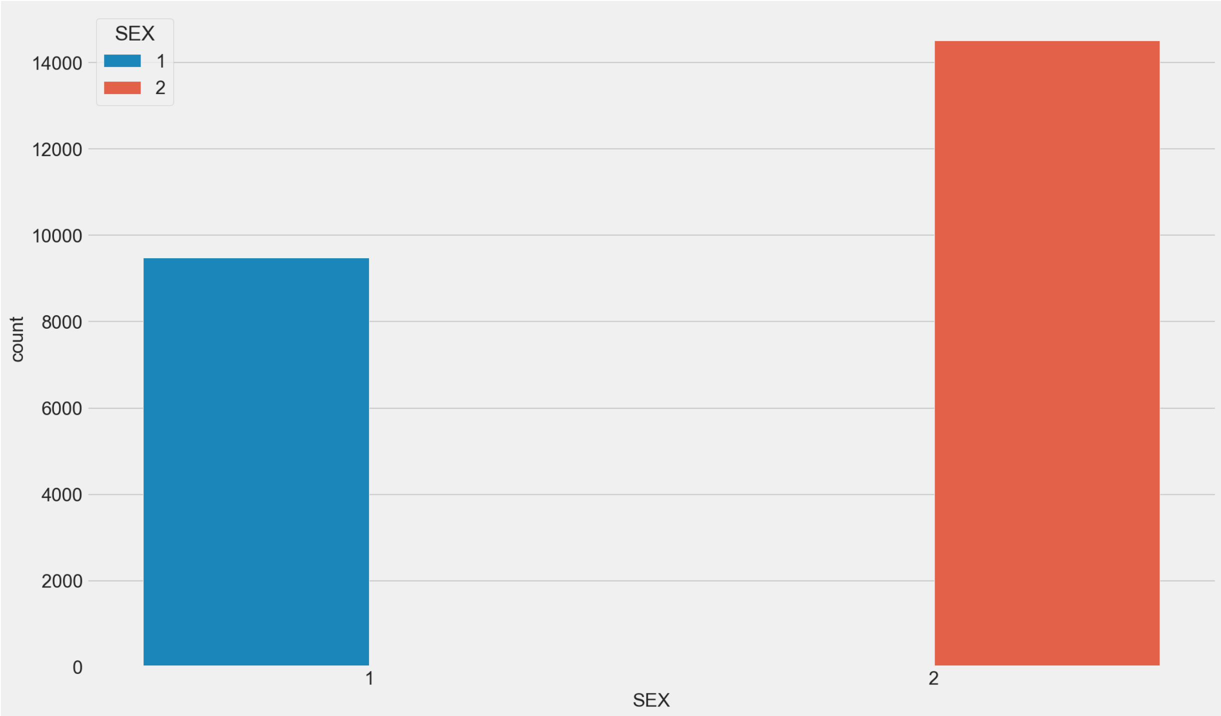
In [23]:
```
sns.barplot(x='SEX',y='LIMIT_BAL',data=X_train,hue='SEX')
```

Out[23]: <AxesSubplot:xlabel='SEX', ylabel='LIMIT_BAL'>



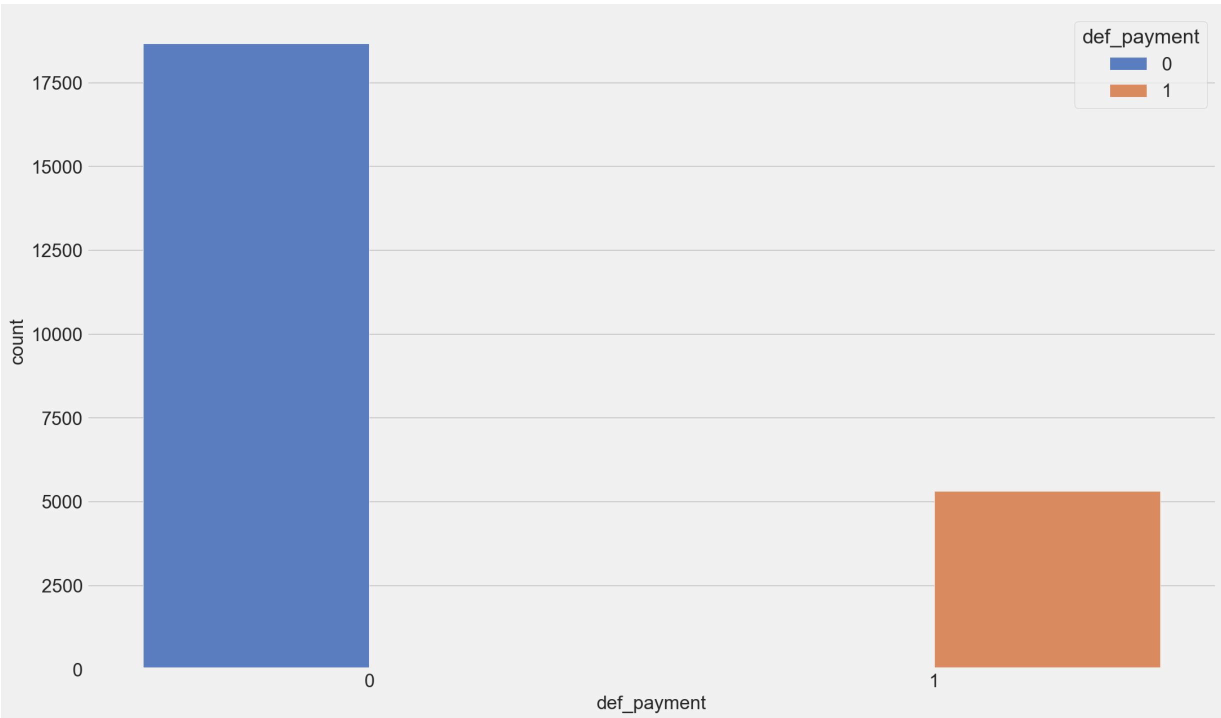
In [24]:
```
sns.countplot(x='SEX',data=X_train,hue='SEX')
```

Out[24]: <AxesSubplot:xlabel='SEX', ylabel='count'>



In [25]:
```
sns.countplot(x='def_payment', data=y_train, hue="def_payment", palette="mute
```

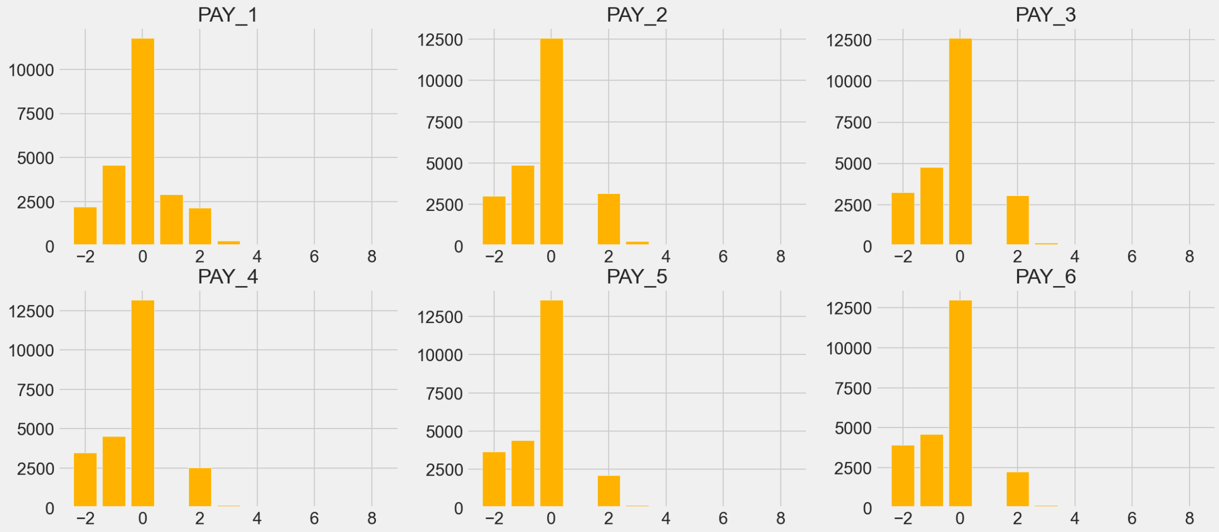
Out[25]: <AxesSubplot:xlabel='def_payment', ylabel='count'>



In [26]:
```
# simple method to plot the features
def getFeatures(prefix):
    return [prefix+str(x) for x in range(1,7)]
```

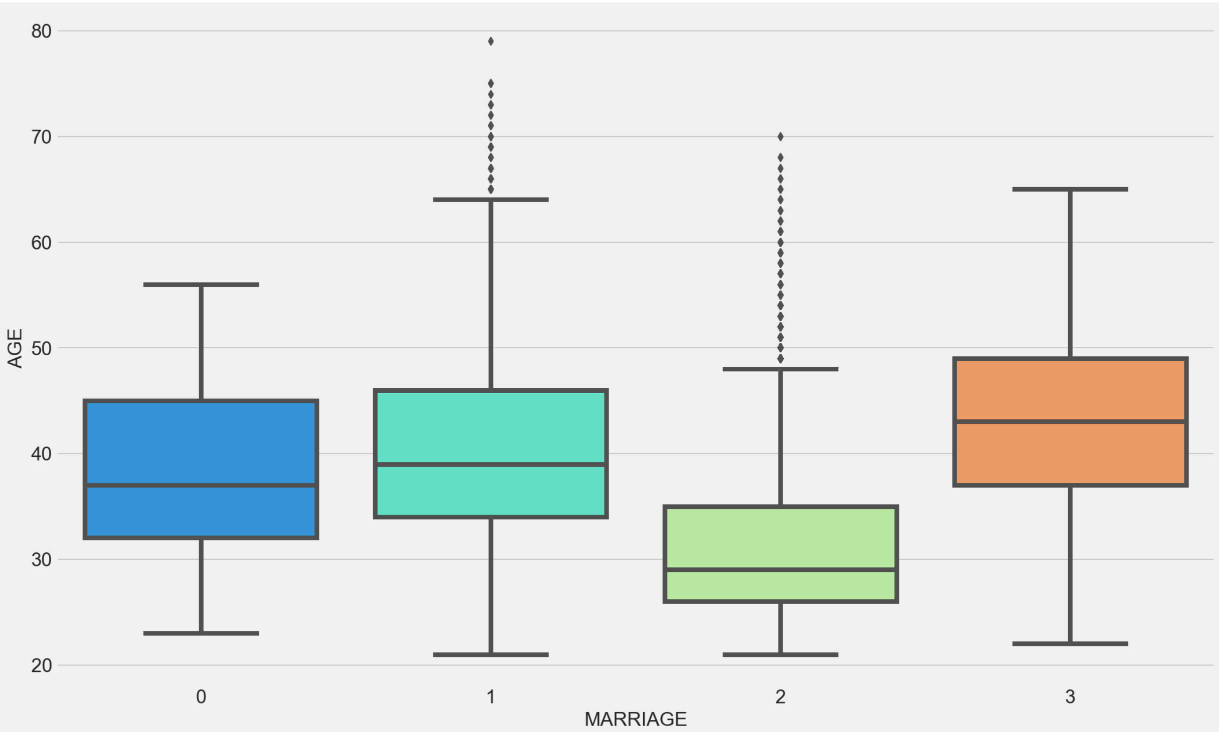
In [27]:
```
pay_status_columns = getFeatures('PAY_')
figure, ax = plt.subplots(2,3)
figure.set_size_inches(18,8)


for i in range(len(pay_status_columns)):
    row,col = int(i/3), i%3

    d  = X_train[pay_status_columns[i]].value_counts()
    x = X_train[pay_status_columns[i]].value_counts()
```

```
    ax[row,col].bar(d.index, d, align='center', color='red')
    ax[row,col].bar(x.index, x, align='center', color='yellow', alpha=0.7)
    ax[row,col].set_title(pay_status_columns[i])



plt.show()
```



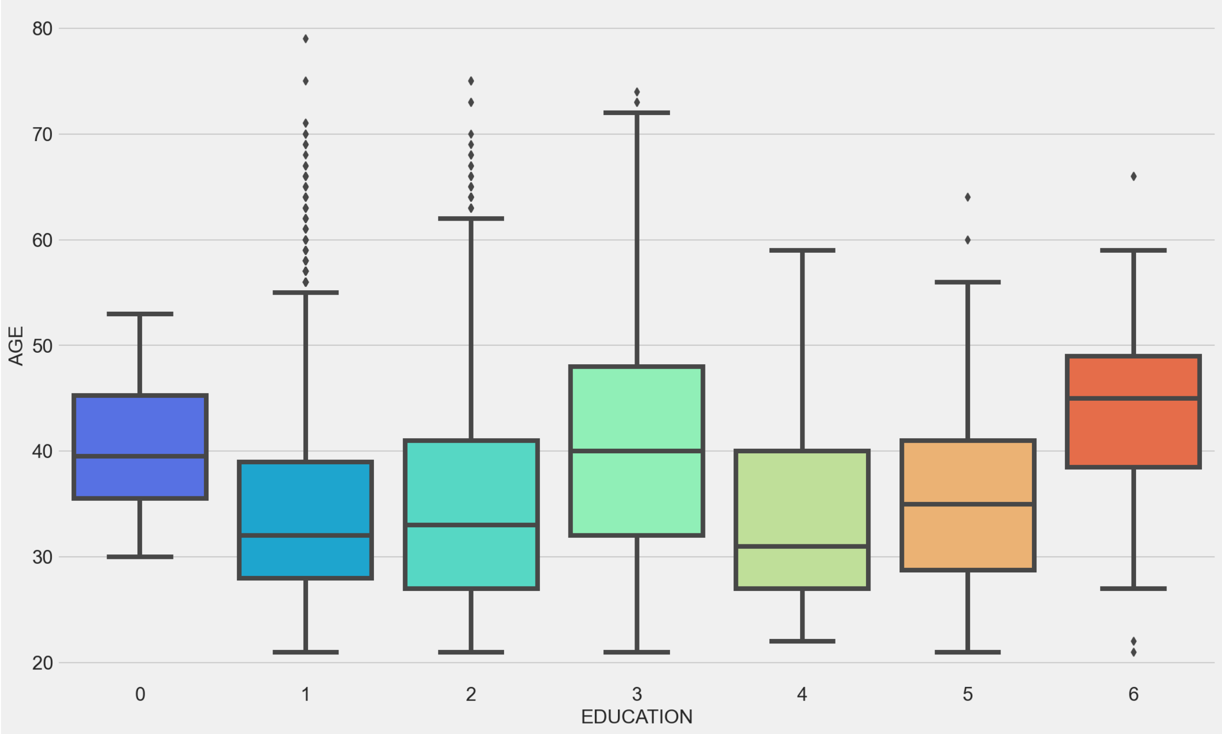
In [28]:
```
sns.boxplot(x='MARRIAGE',y='AGE',data=X_train,palette='rainbow')
```

Out[28]: <AxesSubplot:xlabel='MARRIAGE', ylabel='AGE'>



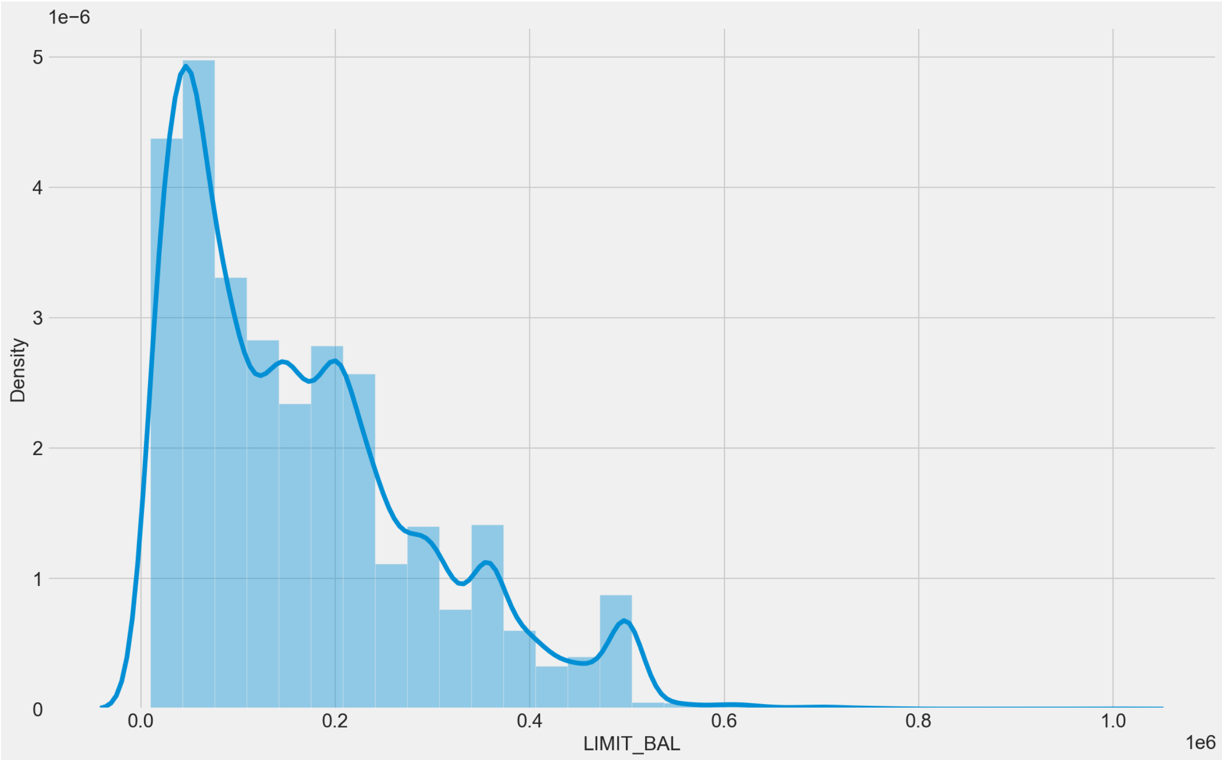
In [29]:
```
sns.boxplot(x='EDUCATION',y='AGE',data=X_train,palette='rainbow')
```

Out[29]: <AxesSubplot:xlabel='EDUCATION', ylabel='AGE'>



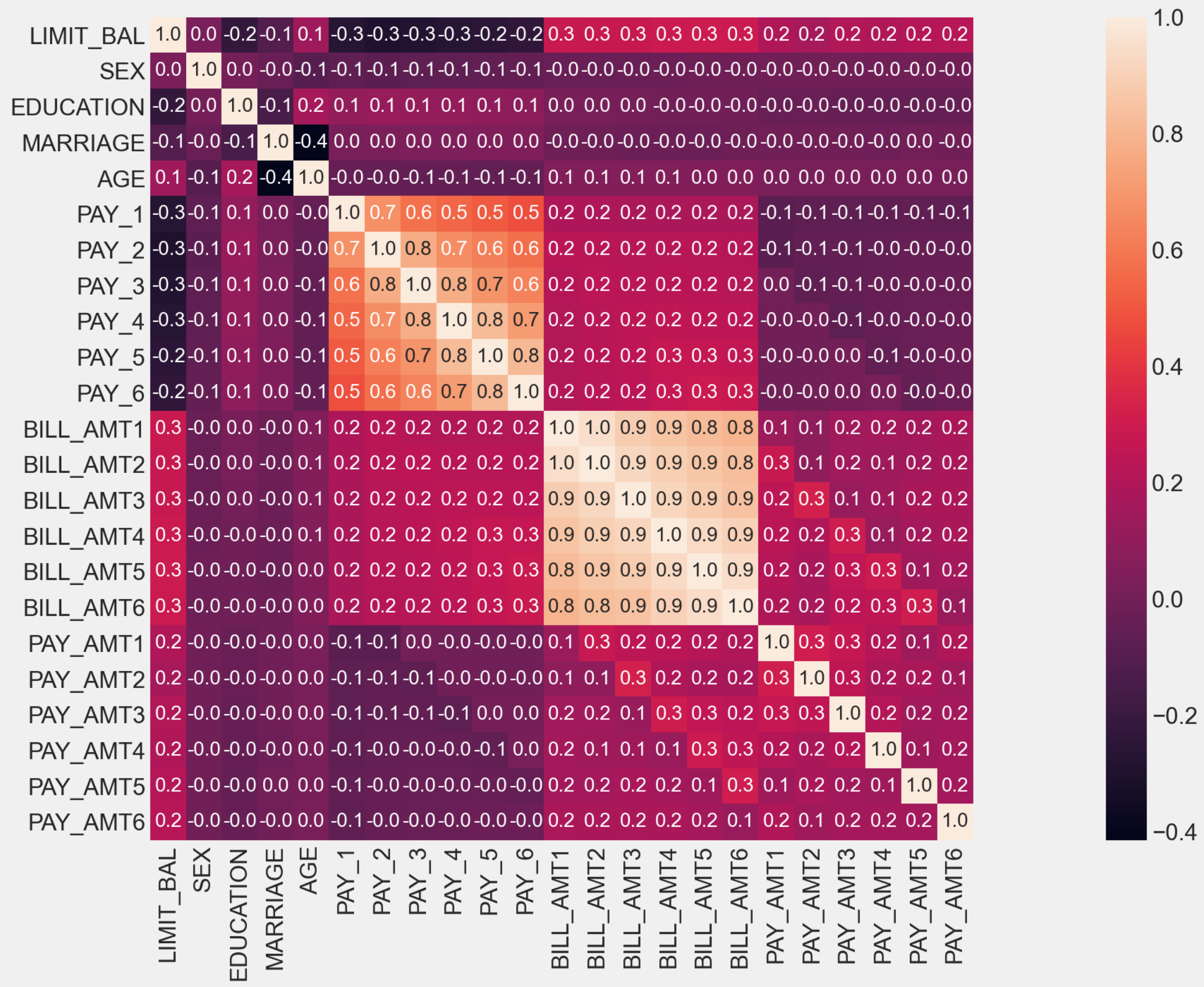
In [30]:
```
sns.distplot(X_train.LIMIT_BAL,kde=True,bins=30)
```

Out[30]: <AxesSubplot:xlabel='LIMIT_BAL', ylabel='Density'>



In [31]:
```
# Obeserving the correlation between features of dataset
correlation = X_train.corr()
plt.subplots(figsize=(30,10))
sns.heatmap( correlation, square=True, annot=True, fmt=".1f" )
```

Out[31]: <AxesSubplot:>



Preprocessing

In [32]:
```
fil = (X_train.EDUCATION == 5) | (X_train.EDUCATION == 6) | (X_train.EDUCATIO
X_train.loc[fil, 'EDUCATION'] = 4
X_train.EDUCATION.value_counts()
```

Out[32]:
```
2    11186
1     8481
3     3959
4      374
Name: EDUCATION, dtype: int64
```

In [33]:
```
fil = (X_test.EDUCATION == 5) | (X_test.EDUCATION == 6) | (X_test.EDUCATION =
X_test.loc[fil, 'EDUCATION'] = 4
X_test.EDUCATION.value_counts()
```

Out[33]:
```
2    2844
1    2104
3     958
4      94
Name: EDUCATION, dtype: int64
```

In [34]:
```
print(X_train['EDUCATION'].value_counts(dropna = False))
print(X_test['EDUCATION'].value_counts(dropna = False))
```

```
2    11186
1     8481
3     3959
4      374
Name: EDUCATION, dtype: int64
```

```
2    2844
1    2104
3     958
4      94
Name: EDUCATION, dtype: int64
```

In [35]:
```python
X_train.loc[X_train.MARRIAGE == 0, 'MARRIAGE'] = 3
X_train.MARRIAGE.value_counts()
```

Out[35]:
```
2    12747
1    10942
3      311
Name: MARRIAGE, dtype: int64
```

In [36]:
```python
X_test.loc[X_test.MARRIAGE == 0, 'MARRIAGE'] = 3
X_test.MARRIAGE.value_counts()
```

Out[36]:
```
2    3217
1    2717
3      66
Name: MARRIAGE, dtype: int64
```

In [37]:
```python
X_train.head()
```

Out[37]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...
21754	80000.0	2	2	2	24	0	0	0	0	0	
252	30000.0	1	2	2	28	0	0	0	0	0	
22942	180000.0	2	4	1	44	0	0	-1	-1	-1	
619	60000.0	1	1	2	25	0	0	0	0	0	
17091	130000.0	2	2	2	25	0	0	0	0	0	

5 rows × 23 columns

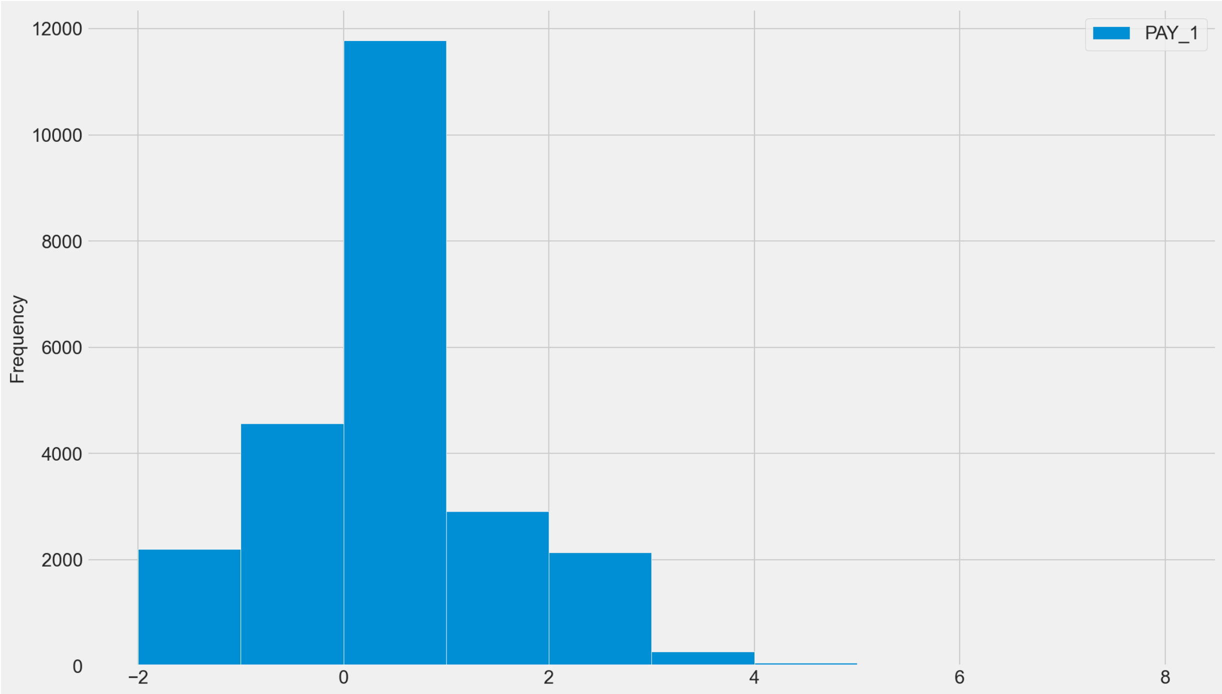
In [38]:
```python
X_train.tail()
```

Out[38]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...
29803	50000.0	1	2	2	32	0	0	0	0	0	
5391	200000.0	1	1	2	37	2	2	2	2	2	
861	50000.0	1	1	2	26	-2	-2	-2	-2	-2	
15796	70000.0	2	2	2	25	0	0	0	0	2	
23655	160000.0	2	2	1	36	-2	-2	-2	-2	-2	

5 rows × 23 columns

In [39]:
```python
X_train.plot(y = 'PAY_1',kind='hist')
plt.legend()
plt.show()
```



In [40]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24000 entries, 21754 to 23655
Data columns (total 23 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   LIMIT_BAL  24000 non-null  float64
 1   SEX        24000 non-null  int64  
 2   EDUCATION  24000 non-null  int64  
 3   MARRIAGE   24000 non-null  int64  
 4   AGE        24000 non-null  int64  
 5   PAY_1      24000 non-null  int64  
 6   PAY_2      24000 non-null  int64  
 7   PAY_3      24000 non-null  int64  
 8   PAY_4      24000 non-null  int64  
 9   PAY_5      24000 non-null  int64  
 10  PAY_6      24000 non-null  int64  
 11  BILL_AMT1  24000 non-null  float64
 12  BILL_AMT2  24000 non-null  float64
 13  BILL_AMT3  24000 non-null  float64
 14  BILL_AMT4  24000 non-null  float64
 15  BILL_AMT5  24000 non-null  float64
 16  BILL_AMT6  24000 non-null  float64
 17  PAY_AMT1   24000 non-null  float64
 18  PAY_AMT2   24000 non-null  float64
 19  PAY_AMT3   24000 non-null  float64
 20  PAY_AMT4   24000 non-null  float64
 21  PAY_AMT5   24000 non-null  float64
 22  PAY_AMT6   24000 non-null  float64
dtypes: float64(13), int64(10)
memory usage: 5.0 MB
```

In [41]:
```
X_train.SEX.nunique()
```

Out[41]: 2

In [42]:
```
X_train[['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT
```

Out[42]:

	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6
count	24000.000000	2.400000e+04	24000.000000	24000.000000	24000.000000	24000.000000
mean	5670.826542	5.961101e+03	5258.246500	4880.847125	4818.849250	5159.462125
std	17084.401034	2.428412e+04	18242.618988	16304.718844	15619.425964	17458.604219
min	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000
25%	1000.000000	8.615000e+02	390.000000	285.750000	240.750000	112.750000
50%	2100.000000	2.007000e+03	1800.000000	1500.000000	1500.000000	1500.000000
75%	5005.000000	5.000000e+03	4500.000000	4000.000000	4021.000000	4000.000000
max	873552.000000	1.684259e+06	896040.000000	621000.000000	426529.000000	527143.000000

In [43]:

```
X_train[['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BI
```

Out[43]:

	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AM
count	24000.000000	24000.000000	2.400000e+04	24000.000000	24000.000000	24000.0000
mean	50927.468417	48914.770500	4.675708e+04	43013.532167	40150.333000	38763.5404
std	73400.840274	70923.493353	6.926506e+04	64069.494705	60635.882129	59281.9868
min	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000	-209051.0000
25%	3537.000000	2989.750000	2.699500e+03	2329.000000	1763.000000	1271.7500
50%	22321.500000	21140.500000	2.005000e+04	19010.000000	18085.000000	17108.5000
75%	66377.000000	63035.250000	5.952925e+04	53927.750000	50007.500000	49101.7500
max	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000	961664.0000

Encoding of the categorical variable

In [44]:

```
categorical_vars = ['SEX','EDUCATION','MARRIAGE','PAY_1','PAY_2','PAY_3','PAY
X_train[categorical_vars].astype(str)
X_test[categorical_vars].astype(str)
X_train.head()
```

Out[44]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	.
ID											
21754	80000.0	2	2	2	24	0	0	0	0	0	
252	30000.0	1	2	2	28	0	0	0	0	0	
22942	180000.0	2	4	1	44	0	0	-1	-1	-1	
619	60000.0	1	1	2	25	0	0	0	0	0	
17091	130000.0	2	2	2	25	0	0	0	0	0	

5 rows × 23 columns

In [45]:

```
X_train.columns = X_train.columns.map(str.lower)
X_test.columns = X_test.columns.map(str.lower)
```

In [46]:
```python
X_train.head()
```

Out[46]:

	limit_bal	sex	education	marriage	age	pay_1	pay_2	pay_3	pay_4	pay_5	...	bill_am
ID												
21754	80000.0	2	2	2	24	0	0	0	0	0	...	78321
252	30000.0	1	2	2	28	0	0	0	0	0	...	29155
22942	180000.0	2	4	1	44	0	0	-1	-1	-1	...	850
619	60000.0	1	1	2	25	0	0	0	0	0	...	38533
17091	130000.0	2	2	2	25	0	0	0	0	0	...	114734

5 rows × 23 columns

Feature scaling

In [47]:
```python
X_train.head(5)
```

Out[47]:

	limit_bal	sex	education	marriage	age	pay_1	pay_2	pay_3	pay_4	pay_5	...	bill_am
ID												
21754	80000.0	2	2	2	24	0	0	0	0	0	...	78321
252	30000.0	1	2	2	28	0	0	0	0	0	...	29155
22942	180000.0	2	4	1	44	0	0	-1	-1	-1	...	850
619	60000.0	1	1	2	25	0	0	0	0	0	...	38533
17091	130000.0	2	2	2	25	0	0	0	0	0	...	114734

5 rows × 23 columns

In [48]:
```python
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.compose import make_column_transformer

X = X_train
y = np.array(y_train.def_payment.values)

transformer = make_column_transformer(
    (MinMaxScaler(), X_train.columns))
transformer.fit(X)
```

Out[48]:
```
ColumnTransformer(transformers=[('minmaxscaler', MinMaxScaler(),
                                 Index(['limit_bal', 'sex', 'education', 'mar
riage', 'age', 'pay_1', 'pay_2',
       'pay_3', 'pay_4', 'pay_5', 'pay_6', 'bill_amt1', 'bill_amt2',
       'bill_amt3', 'bill_amt4', 'bill_amt5', 'bill_amt6', 'pay_amt1',
       'pay_amt2', 'pay_amt3', 'pay_amt4', 'pay_amt5', 'pay_amt6'],
      dtype='object'))])
```

In [49]:
```python
# scaling
X = transformer.transform(X)
```

Splitting the training and test data

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
X_train.shape
```

Out[50]:
```
(19200, 23)
```

Neural Network Models

1. Neural Network with 3 layers

In [51]:
```
# The following method will help us plotting the F1-Score results
def plot_f1(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('F1')
  plt.plot(hist['epoch'], hist['loss'],
           label='Train F1')
  plt.plot(hist['epoch'], hist['val_loss'],
           label = 'Val F1')
  plt.legend()
  plt.show()
```

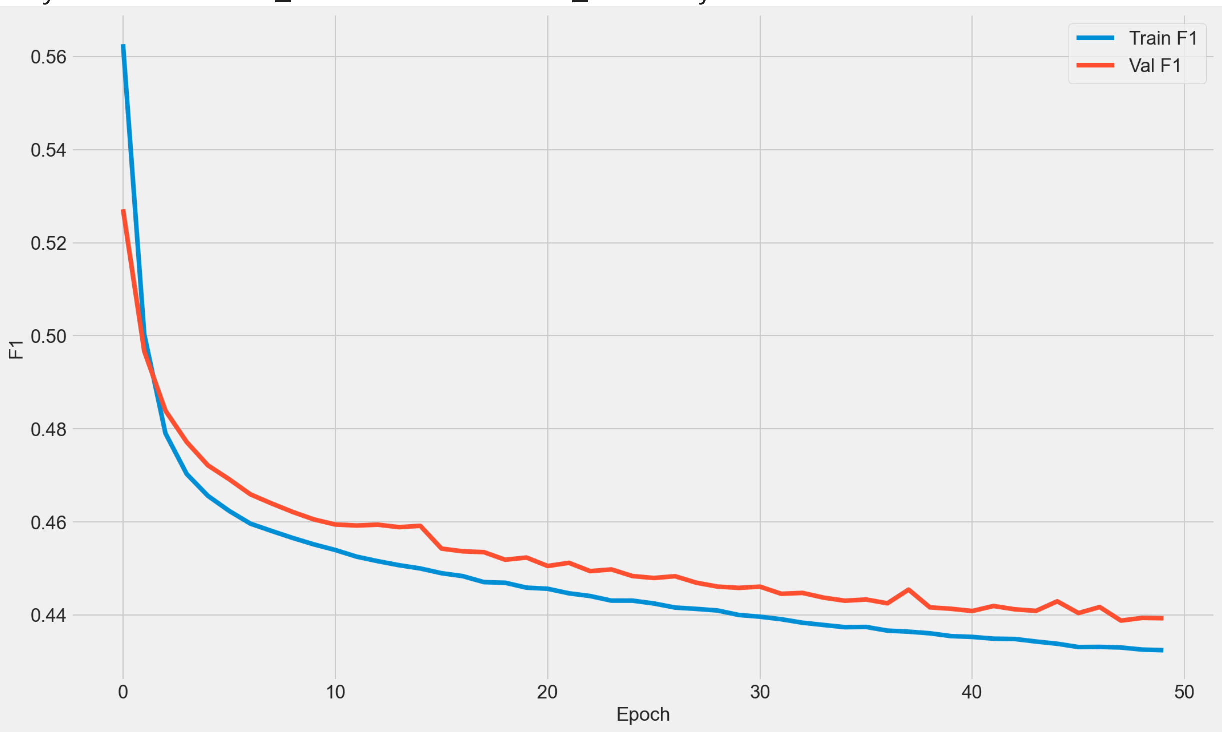
In [52]:
```
model1 = keras.Sequential()
model1.add(keras.layers.Dense(units=32, activation="relu", input_shape=[X_tra
model1.add(keras.layers.Dense(units=64, activation="relu"))
model1.add(keras.layers.Dense(units=128, activation='relu'))

model1.add(keras.layers.Dense(1, activation="sigmoid"))

model1.compile(
    optimizer=keras.optimizers.Adam(0.0001),
    loss = 'binary_crossentropy',
    metrics = ['accuracy'])

BATCH_SIZE = 32

early_stop = keras.callbacks.EarlyStopping(
  monitor='val_loss',
  mode="min",
  patience=10
)

history = model1.fit(
  x=X_train,
  y=y_train,
  shuffle=True,
  epochs=50,
  validation_split=0.2,
  batch_size=BATCH_SIZE
)

plot_f1(history)
```

```
Epoch 1/50
480/480 [==============================] - 1s 3ms/step - loss: 0.5626 - accur
acy: 0.7491 - val_loss: 0.5272 - val_accuracy: 0.7708
Epoch 2/50
480/480 [==============================] - 1s 2ms/step - loss: 0.5004 - accur
acy: 0.7783 - val_loss: 0.4967 - val_accuracy: 0.7737
```

```
Epoch 3/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4790 - accur
acy: 0.7923 - val_loss: 0.4839 - val_accuracy: 0.7896
Epoch 4/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4703 - accur
acy: 0.8025 - val_loss: 0.4772 - val_accuracy: 0.7948
Epoch 5/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4656 - accur
acy: 0.8062 - val_loss: 0.4722 - val_accuracy: 0.8000
Epoch 6/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4624 - accur
acy: 0.8075 - val_loss: 0.4692 - val_accuracy: 0.7995
Epoch 7/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4597 - accur
acy: 0.8094 - val_loss: 0.4659 - val_accuracy: 0.8073
Epoch 8/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4581 - accur
acy: 0.8096 - val_loss: 0.4640 - val_accuracy: 0.8089
Epoch 9/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4566 - accur
acy: 0.8100 - val_loss: 0.4622 - val_accuracy: 0.8091
Epoch 10/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4552 - accur
acy: 0.8098 - val_loss: 0.4606 - val_accuracy: 0.8109
Epoch 11/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4540 - accur
acy: 0.8105 - val_loss: 0.4595 - val_accuracy: 0.8102
Epoch 12/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4526 - accur
acy: 0.8109 - val_loss: 0.4593 - val_accuracy: 0.8107
Epoch 13/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4516 - accur
acy: 0.8120 - val_loss: 0.4594 - val_accuracy: 0.8065
Epoch 14/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4507 - accur
acy: 0.8117 - val_loss: 0.4589 - val_accuracy: 0.8055
Epoch 15/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4500 - accur
acy: 0.8135 - val_loss: 0.4592 - val_accuracy: 0.8044
Epoch 16/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4490 - accur
acy: 0.8133 - val_loss: 0.4543 - val_accuracy: 0.8125
Epoch 17/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4484 - accur
acy: 0.8132 - val_loss: 0.4537 - val_accuracy: 0.8159
Epoch 18/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4471 - accur
acy: 0.8141 - val_loss: 0.4535 - val_accuracy: 0.8151
Epoch 19/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4470 - accur
acy: 0.8139 - val_loss: 0.4519 - val_accuracy: 0.8169
Epoch 20/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4459 - accur
acy: 0.8153 - val_loss: 0.4524 - val_accuracy: 0.8117
Epoch 21/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4457 - accur
acy: 0.8150 - val_loss: 0.4506 - val_accuracy: 0.8182
Epoch 22/50
```

```
480/480 [==============================] - 1s 2ms/step - loss: 0.4447 - accur
acy: 0.8161 - val_loss: 0.4512 - val_accuracy: 0.8146
Epoch 23/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4441 - accur
acy: 0.8162 - val_loss: 0.4495 - val_accuracy: 0.8180
Epoch 24/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4431 - accur
acy: 0.8163 - val_loss: 0.4498 - val_accuracy: 0.8154
Epoch 25/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4431 - accur
acy: 0.8161 - val_loss: 0.4484 - val_accuracy: 0.8177
Epoch 26/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4425 - accur
acy: 0.8165 - val_loss: 0.4480 - val_accuracy: 0.8172
Epoch 27/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4416 - accur
acy: 0.8172 - val_loss: 0.4484 - val_accuracy: 0.8154
Epoch 28/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4413 - accur
acy: 0.8166 - val_loss: 0.4470 - val_accuracy: 0.8161
Epoch 29/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4410 - accur
acy: 0.8171 - val_loss: 0.4461 - val_accuracy: 0.8174
Epoch 30/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4400 - accur
acy: 0.8181 - val_loss: 0.4459 - val_accuracy: 0.8188
Epoch 31/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4397 - accur
acy: 0.8169 - val_loss: 0.4461 - val_accuracy: 0.8148
Epoch 32/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4391 - accur
acy: 0.8176 - val_loss: 0.4446 - val_accuracy: 0.8195
Epoch 33/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4384 - accur
acy: 0.8192 - val_loss: 0.4448 - val_accuracy: 0.8201
Epoch 34/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4379 - accur
acy: 0.8173 - val_loss: 0.4438 - val_accuracy: 0.8195
Epoch 35/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4374 - accur
acy: 0.8180 - val_loss: 0.4431 - val_accuracy: 0.8203
Epoch 36/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4375 - accur
acy: 0.8179 - val_loss: 0.4434 - val_accuracy: 0.8169
Epoch 37/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4367 - accur
acy: 0.8188 - val_loss: 0.4426 - val_accuracy: 0.8190
Epoch 38/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4365 - accur
acy: 0.8187 - val_loss: 0.4455 - val_accuracy: 0.8151
Epoch 39/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4361 - accur
acy: 0.8180 - val_loss: 0.4417 - val_accuracy: 0.8224
Epoch 40/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4355 - accur
acy: 0.8189 - val_loss: 0.4414 - val_accuracy: 0.8214
Epoch 41/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4353 - accur
```

```
acy: 0.8185 - val_loss: 0.4409 - val_accuracy: 0.8214
Epoch 42/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4350 - accur
acy: 0.8192 - val_loss: 0.4420 - val_accuracy: 0.8167
Epoch 43/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4349 - accur
acy: 0.8190 - val_loss: 0.4413 - val_accuracy: 0.8180
Epoch 44/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4343 - accur
acy: 0.8189 - val_loss: 0.4409 - val_accuracy: 0.8182
Epoch 45/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4339 - accur
acy: 0.8188 - val_loss: 0.4430 - val_accuracy: 0.8164
Epoch 46/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4332 - accur
acy: 0.8191 - val_loss: 0.4405 - val_accuracy: 0.8167
Epoch 47/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4332 - accur
acy: 0.8187 - val_loss: 0.4417 - val_accuracy: 0.8169
Epoch 48/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4331 - accur
acy: 0.8184 - val_loss: 0.4388 - val_accuracy: 0.8201
Epoch 49/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4326 - accur
acy: 0.8186 - val_loss: 0.4394 - val_accuracy: 0.8208
Epoch 50/50
480/480 [==============================] - 1s 2ms/step - loss: 0.4325 - accur
acy: 0.8189 - val_loss: 0.4393 - val_accuracy: 0.8201
```



2. Neural Network with SGD Optimizer (4-layers)

In [53]:
```
model2 = keras.Sequential()
model2.add(keras.layers.Dense(units=32, activation="relu", input_shape=[X_tra
model2.add(keras.layers.Dense(units=64, activation="selu"))
model2.add(keras.layers.Dense(units=128, activation="selu"))
model2.add(keras.layers.Dense(units=256, activation="relu"))
model2.add(keras.layers.Dense(1, activation='sigmoid'))
```

```python
model2.compile(
    optimizer=keras.optimizers.SGD(0.0001),
    loss='binary_crossentropy',
    metrics = ['accuracy'])

BATCH_SIZE = 64

early_stop = keras.callbacks.EarlyStopping(
  monitor='val_loss',
  mode="min",
  patience=10
)

history = model2.fit(
  x=X_train,
  y=y_train,
  shuffle=True,
  epochs=100,
  validation_split=0.2,
  batch_size=BATCH_SIZE
)

plot_f1(history)
```

```
Epoch 1/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6763 - accur
acy: 0.7178 - val_loss: 0.6675 - val_accuracy: 0.7555
Epoch 2/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6588 - accur
acy: 0.7710 - val_loss: 0.6516 - val_accuracy: 0.7688
Epoch 3/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6434 - accur
acy: 0.7779 - val_loss: 0.6375 - val_accuracy: 0.7708
Epoch 4/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6297 - accur
acy: 0.7780 - val_loss: 0.6250 - val_accuracy: 0.7708
Epoch 5/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6176 - accur
acy: 0.7781 - val_loss: 0.6141 - val_accuracy: 0.7708
Epoch 6/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6069 - accur
acy: 0.7781 - val_loss: 0.6044 - val_accuracy: 0.7708
Epoch 7/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5974 - accur
acy: 0.7781 - val_loss: 0.5959 - val_accuracy: 0.7708
Epoch 8/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5890 - accur
acy: 0.7781 - val_loss: 0.5884 - val_accuracy: 0.7708
Epoch 9/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5816 - accur
acy: 0.7781 - val_loss: 0.5818 - val_accuracy: 0.7708
Epoch 10/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5751 - accur
acy: 0.7781 - val_loss: 0.5761 - val_accuracy: 0.7708
Epoch 11/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5693 - accur
acy: 0.7781 - val_loss: 0.5710 - val_accuracy: 0.7708
Epoch 12/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5642 - accur
acy: 0.7781 - val_loss: 0.5666 - val_accuracy: 0.7708
```

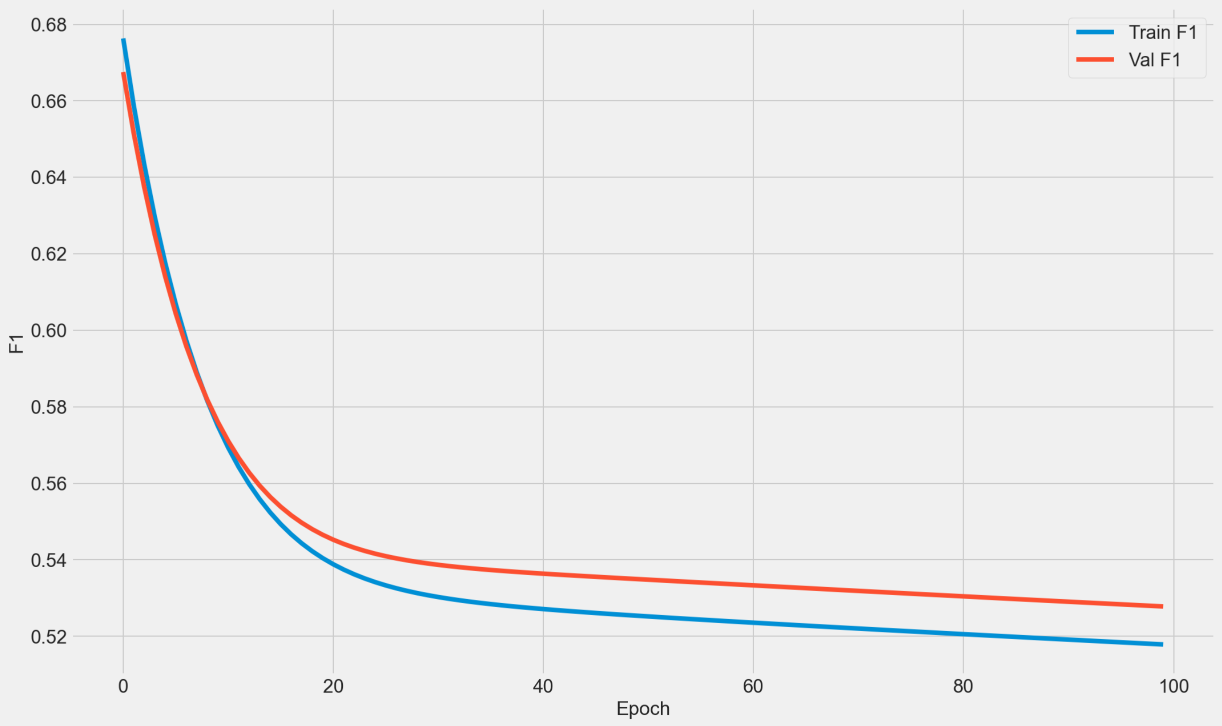
```
Epoch 13/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5598 - accur
acy: 0.7781 - val_loss: 0.5627 - val_accuracy: 0.7708
Epoch 14/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5558 - accur
acy: 0.7781 - val_loss: 0.5594 - val_accuracy: 0.7708
Epoch 15/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5524 - accur
acy: 0.7781 - val_loss: 0.5564 - val_accuracy: 0.7708
Epoch 16/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5493 - accur
acy: 0.7781 - val_loss: 0.5539 - val_accuracy: 0.7708
Epoch 17/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5466 - accur
acy: 0.7781 - val_loss: 0.5516 - val_accuracy: 0.7708
Epoch 18/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5443 - accur
acy: 0.7781 - val_loss: 0.5497 - val_accuracy: 0.7708
Epoch 19/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5422 - accur
acy: 0.7781 - val_loss: 0.5480 - val_accuracy: 0.7708
Epoch 20/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5404 - accur
acy: 0.7781 - val_loss: 0.5465 - val_accuracy: 0.7708
Epoch 21/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5388 - accur
acy: 0.7781 - val_loss: 0.5452 - val_accuracy: 0.7708
Epoch 22/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5374 - accur
acy: 0.7781 - val_loss: 0.5441 - val_accuracy: 0.7708
Epoch 23/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5362 - accur
acy: 0.7781 - val_loss: 0.5432 - val_accuracy: 0.7708
Epoch 24/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5351 - accur
acy: 0.7781 - val_loss: 0.5423 - val_accuracy: 0.7708
Epoch 25/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5342 - accur
acy: 0.7781 - val_loss: 0.5416 - val_accuracy: 0.7708
Epoch 26/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5333 - accur
acy: 0.7781 - val_loss: 0.5409 - val_accuracy: 0.7708
Epoch 27/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5326 - accur
acy: 0.7781 - val_loss: 0.5404 - val_accuracy: 0.7708
Epoch 28/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5319 - accur
acy: 0.7781 - val_loss: 0.5399 - val_accuracy: 0.7708
Epoch 29/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5313 - accur
acy: 0.7781 - val_loss: 0.5394 - val_accuracy: 0.7708
Epoch 30/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5307 - accur
acy: 0.7781 - val_loss: 0.5390 - val_accuracy: 0.7708
Epoch 31/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5303 - accur
acy: 0.7781 - val_loss: 0.5387 - val_accuracy: 0.7708
Epoch 32/100
```

```
240/240 [==============================] - 1s 3ms/step - loss: 0.5298 - accur
acy: 0.7781 - val_loss: 0.5384 - val_accuracy: 0.7708
Epoch 33/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5294 - accur
acy: 0.7781 - val_loss: 0.5381 - val_accuracy: 0.7708
Epoch 34/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5291 - accur
acy: 0.7781 - val_loss: 0.5378 - val_accuracy: 0.7708
Epoch 35/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5287 - accur
acy: 0.7781 - val_loss: 0.5376 - val_accuracy: 0.7708
Epoch 36/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5284 - accur
acy: 0.7781 - val_loss: 0.5373 - val_accuracy: 0.7708
Epoch 37/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5281 - accur
acy: 0.7781 - val_loss: 0.5371 - val_accuracy: 0.7708
Epoch 38/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5279 - accur
acy: 0.7781 - val_loss: 0.5369 - val_accuracy: 0.7708
Epoch 39/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5276 - accur
acy: 0.7781 - val_loss: 0.5367 - val_accuracy: 0.7708
Epoch 40/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5274 - accur
acy: 0.7781 - val_loss: 0.5366 - val_accuracy: 0.7708
Epoch 41/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5271 - accur
acy: 0.7781 - val_loss: 0.5364 - val_accuracy: 0.7708
Epoch 42/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5269 - accur
acy: 0.7781 - val_loss: 0.5362 - val_accuracy: 0.7708
Epoch 43/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5267 - accur
acy: 0.7781 - val_loss: 0.5360 - val_accuracy: 0.7708
Epoch 44/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5265 - accur
acy: 0.7781 - val_loss: 0.5359 - val_accuracy: 0.7708
Epoch 45/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5263 - accur
acy: 0.7781 - val_loss: 0.5357 - val_accuracy: 0.7708
Epoch 46/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5261 - accur
acy: 0.7781 - val_loss: 0.5356 - val_accuracy: 0.7708
Epoch 47/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5259 - accur
acy: 0.7781 - val_loss: 0.5354 - val_accuracy: 0.7708
Epoch 48/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5257 - accur
acy: 0.7781 - val_loss: 0.5353 - val_accuracy: 0.7708
Epoch 49/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5255 - accur
acy: 0.7781 - val_loss: 0.5351 - val_accuracy: 0.7708
Epoch 50/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5254 - accur
acy: 0.7781 - val_loss: 0.5350 - val_accuracy: 0.7708
Epoch 51/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5252 - accur
```

```
acy: 0.7781 - val_loss: 0.5348 - val_accuracy: 0.7708
Epoch 52/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5250 - accur
acy: 0.7781 - val_loss: 0.5347 - val_accuracy: 0.7708
Epoch 53/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5249 - accur
acy: 0.7781 - val_loss: 0.5345 - val_accuracy: 0.7708
Epoch 54/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5247 - accur
acy: 0.7781 - val_loss: 0.5344 - val_accuracy: 0.7708
Epoch 55/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5245 - accur
acy: 0.7781 - val_loss: 0.5342 - val_accuracy: 0.7708
Epoch 56/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5244 - accur
acy: 0.7781 - val_loss: 0.5341 - val_accuracy: 0.7708
Epoch 57/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5242 - accur
acy: 0.7781 - val_loss: 0.5339 - val_accuracy: 0.7708
Epoch 58/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5240 - accur
acy: 0.7781 - val_loss: 0.5338 - val_accuracy: 0.7708
Epoch 59/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5239 - accur
acy: 0.7781 - val_loss: 0.5336 - val_accuracy: 0.7708
Epoch 60/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5237 - accur
acy: 0.7781 - val_loss: 0.5335 - val_accuracy: 0.7708
Epoch 61/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5236 - accur
acy: 0.7781 - val_loss: 0.5333 - val_accuracy: 0.7708
Epoch 62/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5234 - accur
acy: 0.7781 - val_loss: 0.5332 - val_accuracy: 0.7708
Epoch 63/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5232 - accur
acy: 0.7781 - val_loss: 0.5330 - val_accuracy: 0.7708
Epoch 64/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5231 - accur
acy: 0.7781 - val_loss: 0.5329 - val_accuracy: 0.7708
Epoch 65/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5229 - accur
acy: 0.7781 - val_loss: 0.5327 - val_accuracy: 0.7708
Epoch 66/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5228 - accur
acy: 0.7781 - val_loss: 0.5326 - val_accuracy: 0.7708
Epoch 67/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5226 - accur
acy: 0.7781 - val_loss: 0.5325 - val_accuracy: 0.7708
Epoch 68/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5225 - accur
acy: 0.7781 - val_loss: 0.5323 - val_accuracy: 0.7708
Epoch 69/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5223 - accur
acy: 0.7781 - val_loss: 0.5322 - val_accuracy: 0.7708
Epoch 70/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5222 - accur
acy: 0.7781 - val_loss: 0.5320 - val_accuracy: 0.7708
```

```
Epoch 71/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5220 - accur
acy: 0.7781 - val_loss: 0.5319 - val_accuracy: 0.7708
Epoch 72/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5219 - accur
acy: 0.7781 - val_loss: 0.5317 - val_accuracy: 0.7708
Epoch 73/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5217 - accur
acy: 0.7781 - val_loss: 0.5316 - val_accuracy: 0.7708
Epoch 74/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5216 - accur
acy: 0.7781 - val_loss: 0.5314 - val_accuracy: 0.7708
Epoch 75/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5214 - accur
acy: 0.7781 - val_loss: 0.5313 - val_accuracy: 0.7708
Epoch 76/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5213 - accur
acy: 0.7781 - val_loss: 0.5312 - val_accuracy: 0.7708
Epoch 77/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5211 - accur
acy: 0.7781 - val_loss: 0.5310 - val_accuracy: 0.7708
Epoch 78/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5210 - accur
acy: 0.7781 - val_loss: 0.5309 - val_accuracy: 0.7708
Epoch 79/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5208 - accur
acy: 0.7781 - val_loss: 0.5307 - val_accuracy: 0.7708
Epoch 80/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5207 - accur
acy: 0.7781 - val_loss: 0.5306 - val_accuracy: 0.7708
Epoch 81/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5206 - accur
acy: 0.7781 - val_loss: 0.5304 - val_accuracy: 0.7708
Epoch 82/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5204 - accur
acy: 0.7781 - val_loss: 0.5303 - val_accuracy: 0.7708
Epoch 83/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5203 - accur
acy: 0.7781 - val_loss: 0.5302 - val_accuracy: 0.7708
Epoch 84/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5201 - accur
acy: 0.7781 - val_loss: 0.5300 - val_accuracy: 0.7708
Epoch 85/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5200 - accur
acy: 0.7781 - val_loss: 0.5299 - val_accuracy: 0.7708
Epoch 86/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5198 - accur
acy: 0.7781 - val_loss: 0.5297 - val_accuracy: 0.7708
Epoch 87/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5197 - accur
acy: 0.7781 - val_loss: 0.5296 - val_accuracy: 0.7708
Epoch 88/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5196 - accur
acy: 0.7781 - val_loss: 0.5295 - val_accuracy: 0.7708
Epoch 89/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5194 - accur
acy: 0.7781 - val_loss: 0.5293 - val_accuracy: 0.7708
Epoch 90/100
```

```
240/240 [==============================] - 1s 3ms/step - loss: 0.5193 - accur
acy: 0.7781 - val_loss: 0.5292 - val_accuracy: 0.7708
Epoch 91/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5191 - accur
acy: 0.7781 - val_loss: 0.5290 - val_accuracy: 0.7708
Epoch 92/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5190 - accur
acy: 0.7781 - val_loss: 0.5289 - val_accuracy: 0.7708
Epoch 93/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5189 - accur
acy: 0.7781 - val_loss: 0.5288 - val_accuracy: 0.7708
Epoch 94/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5187 - accur
acy: 0.7781 - val_loss: 0.5286 - val_accuracy: 0.7708
Epoch 95/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5186 - accur
acy: 0.7781 - val_loss: 0.5285 - val_accuracy: 0.7708
Epoch 96/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5184 - accur
acy: 0.7781 - val_loss: 0.5284 - val_accuracy: 0.7708
Epoch 97/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5183 - accur
acy: 0.7781 - val_loss: 0.5282 - val_accuracy: 0.7708
Epoch 98/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5181 - accur
acy: 0.7781 - val_loss: 0.5281 - val_accuracy: 0.7708
Epoch 99/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5180 - accur
acy: 0.7781 - val_loss: 0.5279 - val_accuracy: 0.7708
Epoch 100/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5179 - accur
acy: 0.7781 - val_loss: 0.5278 - val_accuracy: 0.7708
```



3. Neural Network with 4 layers and Adagrad Optimizer

In [84]:
```
model3 = keras.Sequential()
model3.add(keras.layers.Dense(units=64, activation="relu", input_shape=[X_tra
```

```python
model3.add(keras.layers.Dense(units=128, activation="linear"))
model3.add(keras.layers.Dense(units=256, activation="selu"))
model3.add(keras.layers.Dense(units=512, activation="relu"))
model3.add(keras.layers.Dense(1, activation='sigmoid'))

model3.compile(
    optimizer=keras.optimizers.Adagrad(0.001),
    loss='binary_crossentropy',
    metrics = ['accuracy'])

BATCH_SIZE = 64

early_stop = keras.callbacks.EarlyStopping(
  monitor='val_loss',
  mode="min",
  patience=10
)

history = model3.fit(
  x=X_train,
  y=y_train,
  shuffle=True,
  epochs=100,
  validation_split=0.2,
  batch_size=BATCH_SIZE
)

plot_f1(history)
```

```
Epoch 1/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5763 - accur
acy: 0.7781 - val_loss: 0.5497 - val_accuracy: 0.7708
Epoch 2/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5367 - accur
acy: 0.7781 - val_loss: 0.5424 - val_accuracy: 0.7708
Epoch 3/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5307 - accur
acy: 0.7781 - val_loss: 0.5374 - val_accuracy: 0.7708
Epoch 4/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5258 - accur
acy: 0.7781 - val_loss: 0.5327 - val_accuracy: 0.7708
Epoch 5/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5211 - accur
acy: 0.7781 - val_loss: 0.5280 - val_accuracy: 0.7708
Epoch 6/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5165 - accur
acy: 0.7781 - val_loss: 0.5232 - val_accuracy: 0.7708
Epoch 7/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5118 - accur
acy: 0.7781 - val_loss: 0.5183 - val_accuracy: 0.7708
Epoch 8/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5071 - accur
acy: 0.7781 - val_loss: 0.5136 - val_accuracy: 0.7708
Epoch 9/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5026 - accur
acy: 0.7781 - val_loss: 0.5092 - val_accuracy: 0.7706
Epoch 10/100
240/240 [==============================] - 1s 2ms/step - loss: 0.4982 - accur
acy: 0.7781 - val_loss: 0.5049 - val_accuracy: 0.7711
Epoch 11/100
240/240 [==============================] - 1s 3ms/step - loss: 0.4941 - accur
```

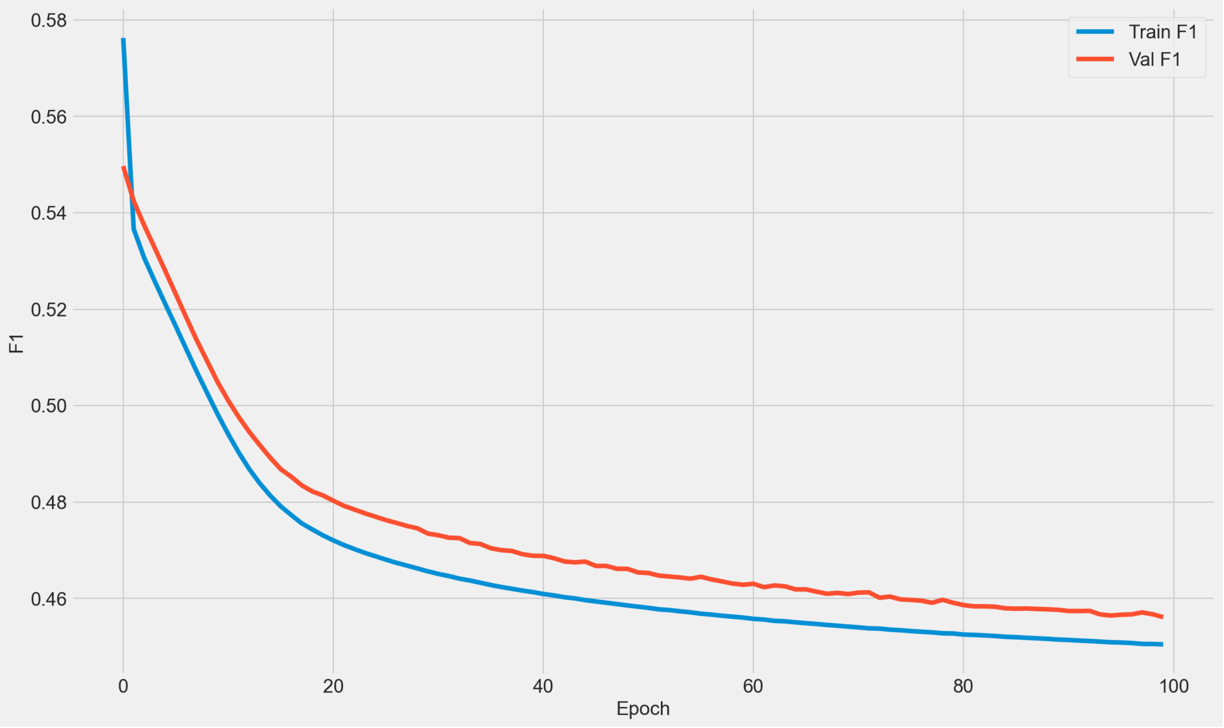
```
acy: 0.7778 - val_loss: 0.5011 - val_accuracy: 0.7716
Epoch 12/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4903 - accur
acy: 0.7783 - val_loss: 0.4977 - val_accuracy: 0.7721
Epoch 13/100
240/240 [==============================] - 2s 6ms/step - loss: 0.4869 - accur
acy: 0.7790 - val_loss: 0.4946 - val_accuracy: 0.7729
Epoch 14/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4839 - accur
acy: 0.7815 - val_loss: 0.4918 - val_accuracy: 0.7747
Epoch 15/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4814 - accur
acy: 0.7842 - val_loss: 0.4892 - val_accuracy: 0.7776
Epoch 16/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4791 - accur
acy: 0.7876 - val_loss: 0.4868 - val_accuracy: 0.7818
Epoch 17/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4773 - accur
acy: 0.7915 - val_loss: 0.4853 - val_accuracy: 0.7849
Epoch 18/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4756 - accur
acy: 0.7933 - val_loss: 0.4835 - val_accuracy: 0.7893
Epoch 19/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4744 - accur
acy: 0.7967 - val_loss: 0.4822 - val_accuracy: 0.7917
Epoch 20/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4731 - accur
acy: 0.7994 - val_loss: 0.4814 - val_accuracy: 0.7914
Epoch 21/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4721 - accur
acy: 0.8008 - val_loss: 0.4803 - val_accuracy: 0.7937
Epoch 22/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4711 - accur
acy: 0.8010 - val_loss: 0.4792 - val_accuracy: 0.7958
Epoch 23/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4703 - accur
acy: 0.8021 - val_loss: 0.4785 - val_accuracy: 0.7958
Epoch 24/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4695 - accur
acy: 0.8025 - val_loss: 0.4777 - val_accuracy: 0.7964
Epoch 25/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4688 - accur
acy: 0.8027 - val_loss: 0.4770 - val_accuracy: 0.7977
Epoch 26/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4681 - accur
acy: 0.8037 - val_loss: 0.4763 - val_accuracy: 0.7982
Epoch 27/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4674 - accur
acy: 0.8042 - val_loss: 0.4757 - val_accuracy: 0.7979
Epoch 28/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4669 - accur
acy: 0.8046 - val_loss: 0.4751 - val_accuracy: 0.7990
Epoch 29/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4663 - accur
acy: 0.8049 - val_loss: 0.4746 - val_accuracy: 0.7992
Epoch 30/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4657 - accur
acy: 0.8052 - val_loss: 0.4735 - val_accuracy: 0.8016
```

```
Epoch 31/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4651 - accur
acy: 0.8051 - val_loss: 0.4731 - val_accuracy: 0.8021
Epoch 32/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4647 - accur
acy: 0.8052 - val_loss: 0.4726 - val_accuracy: 0.8021
Epoch 33/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4641 - accur
acy: 0.8061 - val_loss: 0.4726 - val_accuracy: 0.8016
Epoch 34/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4638 - accur
acy: 0.8053 - val_loss: 0.4715 - val_accuracy: 0.8026
Epoch 35/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4633 - accur
acy: 0.8064 - val_loss: 0.4713 - val_accuracy: 0.8029
Epoch 36/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4628 - accur
acy: 0.8058 - val_loss: 0.4704 - val_accuracy: 0.8034
Epoch 37/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4624 - accur
acy: 0.8073 - val_loss: 0.4700 - val_accuracy: 0.8029
Epoch 38/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4620 - accur
acy: 0.8066 - val_loss: 0.4699 - val_accuracy: 0.8034
Epoch 39/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4617 - accur
acy: 0.8065 - val_loss: 0.4692 - val_accuracy: 0.8029
Epoch 40/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4613 - accur
acy: 0.8064 - val_loss: 0.4689 - val_accuracy: 0.8034
Epoch 41/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4610 - accur
acy: 0.8069 - val_loss: 0.4689 - val_accuracy: 0.8031
Epoch 42/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4607 - accur
acy: 0.8066 - val_loss: 0.4684 - val_accuracy: 0.8034
Epoch 43/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4603 - accur
acy: 0.8071 - val_loss: 0.4677 - val_accuracy: 0.8044
Epoch 44/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4600 - accur
acy: 0.8070 - val_loss: 0.4675 - val_accuracy: 0.8031
Epoch 45/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4597 - accur
acy: 0.8074 - val_loss: 0.4677 - val_accuracy: 0.8034
Epoch 46/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4594 - accur
acy: 0.8072 - val_loss: 0.4668 - val_accuracy: 0.8052
Epoch 47/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4591 - accur
acy: 0.8079 - val_loss: 0.4668 - val_accuracy: 0.8036
Epoch 48/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4589 - accur
acy: 0.8077 - val_loss: 0.4662 - val_accuracy: 0.8052
Epoch 49/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4586 - accur
acy: 0.8077 - val_loss: 0.4662 - val_accuracy: 0.8044
Epoch 50/100
```

```
240/240 [==============================] - 1s 5ms/step - loss: 0.4583 - accur
acy: 0.8079 - val_loss: 0.4654 - val_accuracy: 0.8047
Epoch 51/100
240/240 [==============================] - 2s 6ms/step - loss: 0.4581 - accur
acy: 0.8075 - val_loss: 0.4653 - val_accuracy: 0.8047
Epoch 52/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4578 - accur
acy: 0.8077 - val_loss: 0.4648 - val_accuracy: 0.8047
Epoch 53/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4576 - accur
acy: 0.8081 - val_loss: 0.4646 - val_accuracy: 0.8052
Epoch 54/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4574 - accur
acy: 0.8083 - val_loss: 0.4644 - val_accuracy: 0.8047
Epoch 55/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4571 - accur
acy: 0.8079 - val_loss: 0.4641 - val_accuracy: 0.8047
Epoch 56/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4569 - accur
acy: 0.8084 - val_loss: 0.4645 - val_accuracy: 0.8047
Epoch 57/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4567 - accur
acy: 0.8083 - val_loss: 0.4640 - val_accuracy: 0.8049
Epoch 58/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4564 - accur
acy: 0.8080 - val_loss: 0.4636 - val_accuracy: 0.8047
Epoch 59/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4562 - accur
acy: 0.8083 - val_loss: 0.4631 - val_accuracy: 0.8049
Epoch 60/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4561 - accur
acy: 0.8087 - val_loss: 0.4629 - val_accuracy: 0.8055
Epoch 61/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4558 - accur
acy: 0.8092 - val_loss: 0.4631 - val_accuracy: 0.8039
Epoch 62/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4556 - accur
acy: 0.8089 - val_loss: 0.4624 - val_accuracy: 0.8057
Epoch 63/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4554 - accur
acy: 0.8094 - val_loss: 0.4627 - val_accuracy: 0.8042
Epoch 64/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4553 - accur
acy: 0.8096 - val_loss: 0.4625 - val_accuracy: 0.8044
Epoch 65/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4551 - accur
acy: 0.8093 - val_loss: 0.4619 - val_accuracy: 0.8065
Epoch 66/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4549 - accur
acy: 0.8090 - val_loss: 0.4619 - val_accuracy: 0.8055
Epoch 67/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4547 - accur
acy: 0.8094 - val_loss: 0.4614 - val_accuracy: 0.8065
Epoch 68/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4545 - accur
acy: 0.8095 - val_loss: 0.4610 - val_accuracy: 0.8057
Epoch 69/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4544 - accur
```

```
acy: 0.8102 - val_loss: 0.4612 - val_accuracy: 0.8068
Epoch 70/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4542 - accur
acy: 0.8098 - val_loss: 0.4609 - val_accuracy: 0.8062
Epoch 71/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4540 - accur
acy: 0.8103 - val_loss: 0.4612 - val_accuracy: 0.8055
Epoch 72/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4538 - accur
acy: 0.8103 - val_loss: 0.4613 - val_accuracy: 0.8044
Epoch 73/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4538 - accur
acy: 0.8098 - val_loss: 0.4602 - val_accuracy: 0.8062
Epoch 74/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4536 - accur
acy: 0.8111 - val_loss: 0.4604 - val_accuracy: 0.8060
Epoch 75/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4534 - accur
acy: 0.8102 - val_loss: 0.4598 - val_accuracy: 0.8062
Epoch 76/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4533 - accur
acy: 0.8102 - val_loss: 0.4597 - val_accuracy: 0.8065
Epoch 77/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4531 - accur
acy: 0.8100 - val_loss: 0.4596 - val_accuracy: 0.8062
Epoch 78/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4530 - accur
acy: 0.8106 - val_loss: 0.4591 - val_accuracy: 0.8049
Epoch 79/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4528 - accur
acy: 0.8111 - val_loss: 0.4597 - val_accuracy: 0.8060
Epoch 80/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4528 - accur
acy: 0.8107 - val_loss: 0.4591 - val_accuracy: 0.8060
Epoch 81/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4525 - accur
acy: 0.8109 - val_loss: 0.4586 - val_accuracy: 0.8052
Epoch 82/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4525 - accur
acy: 0.8111 - val_loss: 0.4584 - val_accuracy: 0.8049
Epoch 83/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4524 - accur
acy: 0.8117 - val_loss: 0.4584 - val_accuracy: 0.8055
Epoch 84/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4522 - accur
acy: 0.8113 - val_loss: 0.4583 - val_accuracy: 0.8055
Epoch 85/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4521 - accur
acy: 0.8117 - val_loss: 0.4580 - val_accuracy: 0.8049
Epoch 86/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4520 - accur
acy: 0.8116 - val_loss: 0.4579 - val_accuracy: 0.8060
Epoch 87/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4519 - accur
acy: 0.8118 - val_loss: 0.4579 - val_accuracy: 0.8057
Epoch 88/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4518 - accur
acy: 0.8115 - val_loss: 0.4578 - val_accuracy: 0.8060
```

```
Epoch 89/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4516 - accur
acy: 0.8117 - val_loss: 0.4578 - val_accuracy: 0.8060
Epoch 90/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4515 - accur
acy: 0.8116 - val_loss: 0.4577 - val_accuracy: 0.8057
Epoch 91/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4514 - accur
acy: 0.8120 - val_loss: 0.4574 - val_accuracy: 0.8060
Epoch 92/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4513 - accur
acy: 0.8122 - val_loss: 0.4574 - val_accuracy: 0.8060
Epoch 93/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4512 - accur
acy: 0.8123 - val_loss: 0.4575 - val_accuracy: 0.8062
Epoch 94/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4511 - accur
acy: 0.8123 - val_loss: 0.4567 - val_accuracy: 0.8068
Epoch 95/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4510 - accur
acy: 0.8120 - val_loss: 0.4565 - val_accuracy: 0.8065
Epoch 96/100
240/240 [==============================] - 1s 6ms/step - loss: 0.4509 - accur
acy: 0.8133 - val_loss: 0.4566 - val_accuracy: 0.8065
Epoch 97/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4508 - accur
acy: 0.8126 - val_loss: 0.4567 - val_accuracy: 0.8065
Epoch 98/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4506 - accur
acy: 0.8127 - val_loss: 0.4571 - val_accuracy: 0.8062
Epoch 99/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4506 - accur
acy: 0.8123 - val_loss: 0.4568 - val_accuracy: 0.8057
Epoch 100/100
240/240 [==============================] - 1s 5ms/step - loss: 0.4505 - accur
acy: 0.8123 - val_loss: 0.4561 - val_accuracy: 0.8076
```



4. Neural Network with dropout regularization at 30%

In [91]:

```
model4 = keras.Sequential()
model4.add(keras.layers.Dropout(0.3, input_shape=(X_train.shape[1],)))
model4.add(keras.layers.Dense(units=128, activation="relu"))
model4.add(keras.layers.Dropout(0.3))
model4.add(keras.layers.Dense(units=256, activation="relu"))
model4.add(keras.layers.Dropout(0.3))
model4.add(keras.layers.Dense(units=512, activation="relu"))
model4.add(keras.layers.Dropout(0.3))
model4.add(keras.layers.Dense(1, activation='sigmoid'))

model4.compile(
    optimizer=keras.optimizers.Adadelta(0.001),
    loss = 'binary_crossentropy',
    metrics = ['accuracy'])

BATCH_SIZE = 64

early_stop = keras.callbacks.EarlyStopping(
  monitor='val_loss',
  mode="min",
  patience=15
)

history = model4.fit(
  x=X_train,
  y=y_train,
  shuffle=True,
  epochs=100,
  validation_split=0.2,
  batch_size=BATCH_SIZE
)

plot_f1(history)
```

```
Epoch 1/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6996 - accur
acy: 0.4583 - val_loss: 0.6938 - val_accuracy: 0.4818
Epoch 2/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6887 - accur
acy: 0.5746 - val_loss: 0.6836 - val_accuracy: 0.6995
Epoch 3/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6779 - accur
acy: 0.6708 - val_loss: 0.6737 - val_accuracy: 0.7708
Epoch 4/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6669 - accur
acy: 0.7342 - val_loss: 0.6639 - val_accuracy: 0.7708
Epoch 5/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6568 - accur
acy: 0.7634 - val_loss: 0.6547 - val_accuracy: 0.7708
Epoch 6/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6469 - accur
acy: 0.7745 - val_loss: 0.6458 - val_accuracy: 0.7708
Epoch 7/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6384 - accur
acy: 0.7775 - val_loss: 0.6374 - val_accuracy: 0.7708
Epoch 8/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6292 - accur
acy: 0.7781 - val_loss: 0.6294 - val_accuracy: 0.7708
Epoch 9/100
```

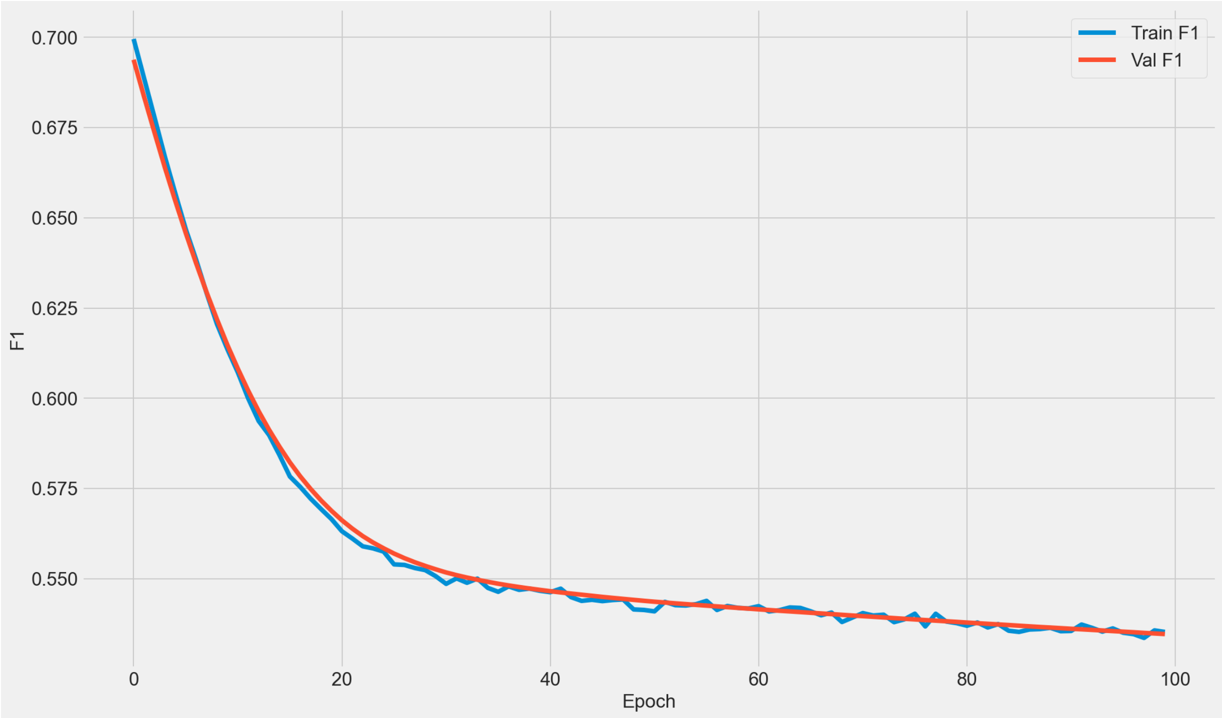
```
240/240 [==============================] - 1s 3ms/step - loss: 0.6206 - accur
acy: 0.7779 - val_loss: 0.6219 - val_accuracy: 0.7708
Epoch 10/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6136 - accur
acy: 0.7781 - val_loss: 0.6148 - val_accuracy: 0.7708
Epoch 11/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6072 - accur
acy: 0.7781 - val_loss: 0.6082 - val_accuracy: 0.7708
Epoch 12/100
240/240 [==============================] - 1s 3ms/step - loss: 0.6000 - accur
acy: 0.7781 - val_loss: 0.6021 - val_accuracy: 0.7708
Epoch 13/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5937 - accur
acy: 0.7781 - val_loss: 0.5964 - val_accuracy: 0.7708
Epoch 14/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5898 - accur
acy: 0.7781 - val_loss: 0.5913 - val_accuracy: 0.7708
Epoch 15/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5843 - accur
acy: 0.7781 - val_loss: 0.5866 - val_accuracy: 0.7708
Epoch 16/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5784 - accur
acy: 0.7781 - val_loss: 0.5823 - val_accuracy: 0.7708
Epoch 17/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5755 - accur
acy: 0.7781 - val_loss: 0.5784 - val_accuracy: 0.7708
Epoch 18/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5722 - accur
acy: 0.7781 - val_loss: 0.5749 - val_accuracy: 0.7708
Epoch 19/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5693 - accur
acy: 0.7781 - val_loss: 0.5717 - val_accuracy: 0.7708
Epoch 20/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5665 - accur
acy: 0.7781 - val_loss: 0.5688 - val_accuracy: 0.7708
Epoch 21/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5632 - accur
acy: 0.7781 - val_loss: 0.5662 - val_accuracy: 0.7708
Epoch 22/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5612 - accur
acy: 0.7781 - val_loss: 0.5639 - val_accuracy: 0.7708
Epoch 23/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5590 - accur
acy: 0.7781 - val_loss: 0.5618 - val_accuracy: 0.7708
Epoch 24/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5584 - accur
acy: 0.7781 - val_loss: 0.5600 - val_accuracy: 0.7708
Epoch 25/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5576 - accur
acy: 0.7781 - val_loss: 0.5585 - val_accuracy: 0.7708
Epoch 26/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5540 - accur
acy: 0.7781 - val_loss: 0.5570 - val_accuracy: 0.7708
Epoch 27/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5539 - accur
acy: 0.7781 - val_loss: 0.5557 - val_accuracy: 0.7708
Epoch 28/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5530 - accur
```

```
acy: 0.7781 - val_loss: 0.5546 - val_accuracy: 0.7708
Epoch 29/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5524 - accur
acy: 0.7781 - val_loss: 0.5536 - val_accuracy: 0.7708
Epoch 30/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5507 - accur
acy: 0.7781 - val_loss: 0.5526 - val_accuracy: 0.7708
Epoch 31/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5486 - accur
acy: 0.7781 - val_loss: 0.5518 - val_accuracy: 0.7708
Epoch 32/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5501 - accur
acy: 0.7781 - val_loss: 0.5510 - val_accuracy: 0.7708
Epoch 33/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5489 - accur
acy: 0.7781 - val_loss: 0.5504 - val_accuracy: 0.7708
Epoch 34/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5501 - accur
acy: 0.7781 - val_loss: 0.5498 - val_accuracy: 0.7708
Epoch 35/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5475 - accur
acy: 0.7781 - val_loss: 0.5492 - val_accuracy: 0.7708
Epoch 36/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5464 - accur
acy: 0.7781 - val_loss: 0.5487 - val_accuracy: 0.7708
Epoch 37/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5479 - accur
acy: 0.7781 - val_loss: 0.5482 - val_accuracy: 0.7708
Epoch 38/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5470 - accur
acy: 0.7781 - val_loss: 0.5477 - val_accuracy: 0.7708
Epoch 39/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5473 - accur
acy: 0.7781 - val_loss: 0.5473 - val_accuracy: 0.7708
Epoch 40/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5467 - accur
acy: 0.7781 - val_loss: 0.5469 - val_accuracy: 0.7708
Epoch 41/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5463 - accur
acy: 0.7781 - val_loss: 0.5466 - val_accuracy: 0.7708
Epoch 42/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5473 - accur
acy: 0.7781 - val_loss: 0.5463 - val_accuracy: 0.7708
Epoch 43/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5449 - accur
acy: 0.7781 - val_loss: 0.5459 - val_accuracy: 0.7708
Epoch 44/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5439 - accur
acy: 0.7781 - val_loss: 0.5456 - val_accuracy: 0.7708
Epoch 45/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5442 - accur
acy: 0.7781 - val_loss: 0.5453 - val_accuracy: 0.7708
Epoch 46/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5438 - accur
acy: 0.7781 - val_loss: 0.5450 - val_accuracy: 0.7708
Epoch 47/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5441 - accur
acy: 0.7781 - val_loss: 0.5447 - val_accuracy: 0.7708
```

```
Epoch 48/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5443 - accur
acy: 0.7781 - val_loss: 0.5445 - val_accuracy: 0.7708
Epoch 49/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5415 - accur
acy: 0.7781 - val_loss: 0.5442 - val_accuracy: 0.7708
Epoch 50/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5414 - accur
acy: 0.7781 - val_loss: 0.5439 - val_accuracy: 0.7708
Epoch 51/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5410 - accur
acy: 0.7781 - val_loss: 0.5437 - val_accuracy: 0.7708
Epoch 52/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5436 - accur
acy: 0.7781 - val_loss: 0.5435 - val_accuracy: 0.7708
Epoch 53/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5427 - accur
acy: 0.7781 - val_loss: 0.5432 - val_accuracy: 0.7708
Epoch 54/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5426 - accur
acy: 0.7781 - val_loss: 0.5430 - val_accuracy: 0.7708
Epoch 55/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5430 - accur
acy: 0.7781 - val_loss: 0.5428 - val_accuracy: 0.7708
Epoch 56/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5439 - accur
acy: 0.7781 - val_loss: 0.5426 - val_accuracy: 0.7708
Epoch 57/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5414 - accur
acy: 0.7781 - val_loss: 0.5424 - val_accuracy: 0.7708
Epoch 58/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5425 - accur
acy: 0.7781 - val_loss: 0.5422 - val_accuracy: 0.7708
Epoch 59/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5420 - accur
acy: 0.7781 - val_loss: 0.5420 - val_accuracy: 0.7708
Epoch 60/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5418 - accur
acy: 0.7781 - val_loss: 0.5418 - val_accuracy: 0.7708
Epoch 61/100
240/240 [==============================] - 1s 2ms/step - loss: 0.5424 - accur
acy: 0.7781 - val_loss: 0.5416 - val_accuracy: 0.7708
Epoch 62/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5410 - accur
acy: 0.7781 - val_loss: 0.5414 - val_accuracy: 0.7708
Epoch 63/100
240/240 [==============================] - 1s 3ms/step - loss: 0.5413 - accur
acy: 0.7781 - val_loss: 0.5412 - val_accuracy: 0.7708
Epoch 64/100
240/240 [==============================] - 1s 6ms/step - loss: 0.5421 - accur
acy: 0.7781 - val_loss: 0.5410 - val_accuracy: 0.7708
Epoch 65/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5420 - accur
acy: 0.7781 - val_loss: 0.5408 - val_accuracy: 0.7708
Epoch 66/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5410 - accur
acy: 0.7781 - val_loss: 0.5406 - val_accuracy: 0.7708
Epoch 67/100
```

```
240/240 [==============================] - 2s 7ms/step - loss: 0.5399 - accur
acy: 0.7781 - val_loss: 0.5404 - val_accuracy: 0.7708
Epoch 68/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5407 - accur
acy: 0.7781 - val_loss: 0.5402 - val_accuracy: 0.7708
Epoch 69/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5381 - accur
acy: 0.7781 - val_loss: 0.5400 - val_accuracy: 0.7708
Epoch 70/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5393 - accur
acy: 0.7781 - val_loss: 0.5399 - val_accuracy: 0.7708
Epoch 71/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5405 - accur
acy: 0.7781 - val_loss: 0.5397 - val_accuracy: 0.7708
Epoch 72/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5399 - accur
acy: 0.7781 - val_loss: 0.5395 - val_accuracy: 0.7708
Epoch 73/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5401 - accur
acy: 0.7781 - val_loss: 0.5393 - val_accuracy: 0.7708
Epoch 74/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5380 - accur
acy: 0.7781 - val_loss: 0.5391 - val_accuracy: 0.7708
Epoch 75/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5388 - accur
acy: 0.7781 - val_loss: 0.5389 - val_accuracy: 0.7708
Epoch 76/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5404 - accur
acy: 0.7781 - val_loss: 0.5388 - val_accuracy: 0.7708
Epoch 77/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5368 - accur
acy: 0.7781 - val_loss: 0.5386 - val_accuracy: 0.7708
Epoch 78/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5403 - accur
acy: 0.7781 - val_loss: 0.5384 - val_accuracy: 0.7708
Epoch 79/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5382 - accur
acy: 0.7781 - val_loss: 0.5383 - val_accuracy: 0.7708
Epoch 80/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5377 - accur
acy: 0.7781 - val_loss: 0.5381 - val_accuracy: 0.7708
Epoch 81/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5370 - accur
acy: 0.7781 - val_loss: 0.5379 - val_accuracy: 0.7708
Epoch 82/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5379 - accur
acy: 0.7781 - val_loss: 0.5377 - val_accuracy: 0.7708
Epoch 83/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5365 - accur
acy: 0.7781 - val_loss: 0.5375 - val_accuracy: 0.7708
Epoch 84/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5375 - accur
acy: 0.7781 - val_loss: 0.5374 - val_accuracy: 0.7708
Epoch 85/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5356 - accur
acy: 0.7781 - val_loss: 0.5372 - val_accuracy: 0.7708
Epoch 86/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5353 - accur
```

```
acy: 0.7781 - val_loss: 0.5370 - val_accuracy: 0.7708
Epoch 87/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5360 - accur
acy: 0.7781 - val_loss: 0.5368 - val_accuracy: 0.7708
Epoch 88/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5361 - accur
acy: 0.7781 - val_loss: 0.5367 - val_accuracy: 0.7708
Epoch 89/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5364 - accur
acy: 0.7781 - val_loss: 0.5365 - val_accuracy: 0.7708
Epoch 90/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5355 - accur
acy: 0.7781 - val_loss: 0.5363 - val_accuracy: 0.7708
Epoch 91/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5356 - accur
acy: 0.7781 - val_loss: 0.5362 - val_accuracy: 0.7708
Epoch 92/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5373 - accur
acy: 0.7781 - val_loss: 0.5360 - val_accuracy: 0.7708
Epoch 93/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5364 - accur
acy: 0.7781 - val_loss: 0.5359 - val_accuracy: 0.7708
Epoch 94/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5354 - accur
acy: 0.7781 - val_loss: 0.5357 - val_accuracy: 0.7708
Epoch 95/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5363 - accur
acy: 0.7781 - val_loss: 0.5355 - val_accuracy: 0.7708
Epoch 96/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5351 - accur
acy: 0.7781 - val_loss: 0.5354 - val_accuracy: 0.7708
Epoch 97/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5347 - accur
acy: 0.7781 - val_loss: 0.5352 - val_accuracy: 0.7708
Epoch 98/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5336 - accur
acy: 0.7781 - val_loss: 0.5350 - val_accuracy: 0.7708
Epoch 99/100
240/240 [==============================] - 2s 6ms/step - loss: 0.5357 - accur
acy: 0.7781 - val_loss: 0.5349 - val_accuracy: 0.7708
Epoch 100/100
240/240 [==============================] - 2s 7ms/step - loss: 0.5353 - accur
acy: 0.7781 - val_loss: 0.5347 - val_accuracy: 0.7708
```



Model Evaluation

In [92]:
```
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall

# We add the predicted score to a file text
f = open("Mirko_Lantieri_858278_score2.txt", "a")
```

In [93]:
```
a = np.asarray(model1.predict(X_test))
f.write(f"{a}\n")
```

Out[93]: 90

In [94]:
```
a = np.asarray(model2.predict(X_test))
f.write(f"{a}\n")
```

Out[94]: 90

In [95]:
```
a = np.asarray(model3.predict(X_test))
f.write(f"{a}\n")
```

Out[95]: 90

In [96]:
```
a = np.asarray(model4.predict(X_test))
f.write(f"{a}\n")
```

Out[96]: 90

In [97]:
```
f.close()
```

Metrics evaluation Model 1

In [98]:
```
y_pred = np.round(model1.predict(X_test))
roc = roc_auc_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
```

```
prec = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

In [99]:
```
results = pd.DataFrame([['Logistic Regression', acc,prec,recall, f1,roc]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Sco
results
```

Out[99]:

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.822083	0.651007	0.375242	0.476074	0.660005

Metrics evaluation Model 2

In [100…
```
y_pred = np.round(model2.predict(X_test))
roc = roc_auc_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

In [86]:
```
results = pd.DataFrame([['Logistic Regression', acc,prec,recall, f1,roc]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Sco
results
```

Out[86]:

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.784583	0.0	0.0	0.0	0.5

Metrics evaluation Model 3

In [101…
```
y_pred = np.round(model3.predict(X_test))
roc = roc_auc_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

In [102…
```
results = pd.DataFrame([['Logistic Regression', acc,prec,recall, f1,roc]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Sco
results
```

Out[102…

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.812292	0.632735	0.306576	0.413029	0.628859

Metrics evaluation Model 4

In [103…
```
y_pred = np.round(model4.predict(X_test))
roc = roc_auc_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

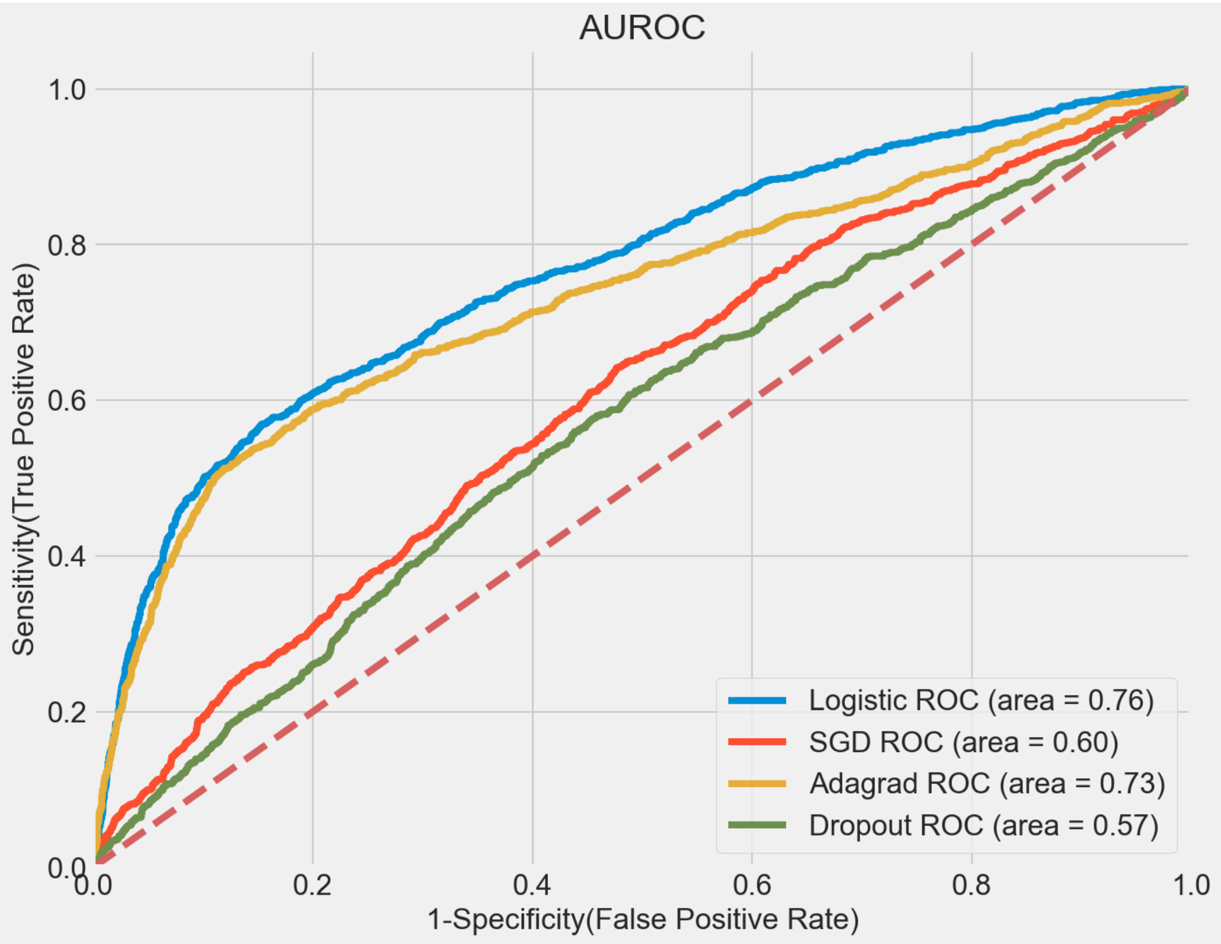
In [104…
```
results = pd.DataFrame([['Logistic Regression', acc,prec,recall, f1,roc]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Sco
results
```

Out[104…

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.784583	0.0	0.0	0.0	0.5

In [105…

```
from sklearn import metrics

# false positive rate,fpr= FP/(TN+FP) OR fpr=1-specificty, tpr=sensitivity
y_pred_1 = model1.predict(X_test)
y_pred_2 = model2.predict(X_test)

y_pred_3 = model3.predict(X_test)
y_pred_4 = model4.predict(X_test)

model = [model1,model2,model3,model4]

models=[y_pred_1,y_pred_2,y_pred_3,y_pred_4]
label=['Logistic','SGD','Adagrad','Dropout']

# plotting ROC curves
plt.figure(figsize=(10, 8))
m=np.arange(4)
for m in m:
    fpr, tpr,thresholds= metrics.roc_curve(y_test,models[m])
    auc = metrics.roc_auc_score(y_test,model[m].predict(X_test))
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (label[m], auc))
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('AUROC')
plt.legend(loc="lower right")
plt.show()
```



In [ ]:

```

```