GDPR_Notebook

May 4, 2022

1 AI challenge

Author: Ivett Elena Fuentes Herrera

2 Description

Sensitive data has great value, which is why many governments have established regulations that address both data protection and data privacy. The General Data Protection Regulation (GDPR) of the European Union (EU) is one of the most influential laws in recent years. Any company that does business in the EU or the European Economic Area (EEA), or that markets products or services for people in the EU or EEA must comply with GDPR standards or face severe financial repercussions in the form of fines and injunctions. The CMS.Law GDPR Enforcement Tracker is an overview of fines and penalties that data protection authorities within the EU have imposed under the EU GDPR.

[1]:
```
from utils import *
```

Several functions are set up in `utils.py`

3 Data loading

Load and summarize the CMS.Law GDPR Enforcement Tracker data, which is an overview of fines and penalties that data protection authorities within the EU have imposed under the EU GDPR

Source: https://www.enforcementtracker.com/

[2]:
```
# load the CMS.Law GDPR data
df = load_data()
# summarize the shape of the dataset
print('The shape is', df.shape)
```

```
The shape is (1147, 11)
```

4 Data representation and pre-processing

Clean and transform raw data into a more understandable, useful, and efficient format

[3]:
```python
# search for missing values
df = df[df['Fine'].notna()]
print('The shape is', df.shape)
df = df.reset_index(drop = True)
# search for Unknown values
unknown_idx = []
for row in range(len(df)):
    art = df.iloc[row,5][0]
    if art == 'Unknown':
        unknown_idx.append(row)
```

```
The shape is (1118, 11)
```

[4]:
```python
df.drop(df.iloc[unknown_idx,:].index, axis=0, inplace=True)
df = df.reset_index(drop=True)
print('The shape is', df.shape)

# search for Fine == 0

del_Fine_idx = df[df.Fine == 0].index
df.drop(del_Fine_idx, axis=0, inplace=True)
print('The shape is', df.shape)

save_dataset(df, "GDPR")
# tail of the dataset
df.tail(5)
```

```
The shape is (1109, 11)
The shape is (1105, 11)
```

[4]:
```
             Id  Country Date_of_decision     Fine  \
1104  ETid-1143    SPAIN       2022-04-29   4200.0
1105  ETid-1144    SPAIN       2022-04-29  16000.0
1106  ETid-1145  ROMANIA       2022-05-03   4000.0
1107  ETid-1146   CYPRUS       2021-09-17  10000.0
1108  ETid-1147    SPAIN       2022-04-28   1500.0

                    Controller_Processor  \
1104  CLÍNICA DENTAL SAN FRANCISCO, S.L.
1105         LABORATORIOS GONZÁLEZ, S.L.
1106              Megareduceri TV S.R.L.
1107    Mediterranean Hospital of Cyprus
1108                 CAFFE VECCHIO, S.L.
```

```
                                   Quoted_Article  \
1104              [Art. 17 GDPR,  Art. 21 LSSI]
1105                        [Art. 5 (1) f) GDPR]
1106                          [Art. 58 (1) GDPR]
1107       [Art. 31 GDPR,  Art. 58 (1) a) GDPR]
1108  [Art. 5 (1) f) GDPR,  Art. 6 (1) a) GDPR]

                                                Type  \
1104   Insufficient fulfilment of data subjects rights
1105  Non-compliance with general data processing pr…
1106  Insufficient cooperation with supervisory auth…
1107  Insufficient cooperation with supervisory auth…
1108      Insufficient legal basis for data processing

                                              Source  \
1104  <a class='blau' href='https://www.aepd.es/es/d…
1105  <a class='blau' href='https://www.aepd.es/es/d…
1106  <a class='blau' href='https://www.dataprotecti…
1107  <a class='blau' href='https://www.dataprotecti…
1108  <a class='blau' href='https://www.aepd.es/es/d…

                                           Authority  \
1104         Spanish Data Protection Authority (aepd)
1105         Spanish Data Protection Authority (aepd)
1106  Romanian National Supervisory Authority for Pe…
1107             Cypriot Data Protection Commissioner
1108         Spanish Data Protection Authority (aepd)

                     Sector                                           Summary
1104            Health Care  The Spanish DPA (AEPD) has imposed a fine on C…
1105            Health Care  The Spanish DPA (AEPD) has fined LABORATORIOS …
1106  Industry and Commerce  Failure to provide requested information to th…
1107            Health Care  The Cypriot DPA (ANSPDCP) has fined Mediterran…
1108             Employment  The Spanish DPA has fined CAFFE VECCHIO, S.L. …
```

5 Feature extraction from structured and unstructured data

Extract useful, structured information from the unstructured data (i.e., *Summary*, *Quoted articles*)

The *Summary* feature provides a free text description of the violation. We transform this feature into numerical features usable for machine learning models by computing TF-IDF, which stands for Term Frequency-Inverse Document Frequency

[5]:
```python
#  tf_idf representation
df = pd.read_csv('Data/GDPR.csv')
df.Summary.replace(regex=r'[^a-zA-Z]', value=' ', inplace=True)
tf_idf = tf_idf_representation_df(df)
```

```
print('The shape is', tf_idf.shape)
save_dataset(tf_idf, "GDPR_Summary_Tf_Idf")
# tail of the dataset
tf_idf.tail(5)
```

```
The shape is (1105, 31)
```

[5]:

```
      access     addit      aepd       art  author  breach  compani   consent  \
1100     0.0  0.000000  0.467931  0.000000     0.0     0.0      0.0  0.000000
1101     0.0  0.000000  0.298954  0.000000     0.0     0.0      0.0  0.338452
1102     0.0  0.000000  0.000000  0.458355     0.0     0.0      0.0  0.000000
1103     0.0  0.000000  0.000000  0.000000     0.0     0.0      0.0  0.000000
1104     0.0  0.380588  0.000000  0.000000     0.0     0.0      0.0  0.000000

      contract   control     …     purpos  receiv   request  secur  \
1100       0.0  0.535264     …        0.0     0.0  0.252572    0.0
1101       0.0  0.000000     …        0.0     0.0  0.000000    0.0
1102       0.0  0.000000     …        0.0     0.0  0.476385    0.0
1103       0.0  0.000000     …        0.0     0.0  0.433148    0.0
1104       0.0  0.000000     …        0.0     0.0  0.000000    0.0

       spanish  surveil  system    violat   Fine         Id
1100  0.233186      0.0     0.0  0.000000   4200  ETid-1143
1101  0.297958      0.0     0.0  0.000000  16000  ETid-1144
1102  0.000000      0.0     0.0  0.424414   4000  ETid-1145
1103  0.000000      0.0     0.0  0.000000  10000  ETid-1146
1104  0.339194      0.0     0.0  0.000000   1500  ETid-1147

[5 rows x 31 columns]
```

Quoted articles, sub-articles, and subsections follow a hierarchical structure that should not be ignored. We should be able to retain article information present in each row. We transform this feature into categorical features usable for machine learning models by computing embeddings. We employ the following approaches: - *Categorical Article representation*, including all descriptions in the *Quoted_article* feature - *Categorical Article representation*, including other articles and only the general category of GDPR quoted article description in the *Quoted_article* feature

[6]:

```
# pre-process the Articles dataframe
article_representation_df = create_article_features_df(df)
# article_representation_df: Dataframe(Id, Article)
article_representation_df.Article.replace(regex='\r\n', value='', inplace=True)
article_representation_df.Article.replace(regex=b'\xa0', value='', inplace=True)
article_representation_df.Article.replace(regex=b'\xa0', value='', inplace=True)
article_representation_df.Article.replace(regex='\n', value='', inplace=True)
article_representation_df.Article.replace(regex=',', value='', inplace=True)
article_representation_df.Article.replace(regex='  ', value=' ', inplace=True)
```

```
article_representation_df.Article.replace(regex='GDPR ', value='GDPR', inplace=True)
article_representation_df.Article.replace(regex='^\s* ', value='', inplace=True)
article_representation_df.Article.replace(regex='$|\s ', value='', inplace=True)

empty_idx = article_representation_df[article_representation_df.Article == ''].index
article_representation_df.drop(empty_idx, axis=0, inplace=True)
article_representation_df = article_representation_df.reset_index(drop=True)
print('The shape is', article_representation_df.shape)
```

```
The shape is (2294, 2)
```

[7]:
```
# Categorical Article representation, including all descriptions in the quoted article feature
article_representation_matrix = categorical_representation_df(article_representation_df,1)
ids_names = pd.DataFrame(list(article_representation_matrix.index.values))
categorical_article_df = merge_dataset(ids_names, 'Data/GDPR_Article_Matrix.csv')
save_dataset(categorical_article_df, "Categorical_GDPR_Article_Df")
print('The shape is', categorical_article_df.shape)
# tail of the dataset
categorical_article_df.tail(5)
```

```
The shape is (1105, 245)
```

[7]:
```
      13 GDPR  5 GDPR  14 GDPR  5 (1) a) GDPR  6 GDPR  5 (1) c) GDPR  \
1100        0       0        0              0       0              0
1101        0       0        0              0       0              0
1102        0       0        0              0       0              0
1103        0       0        0              0       0              0
1104        0       0        0              0       0              0

      6 (1) GDPR  5 (1) b) GDPR  15 GDPR  32 GDPR  …  9 (2) i) GDPR  \
1100           0              0        0        0  …              0
1101           0              0        0        0  …              0
1102           0              0        0        0  …              0
1103           0              0        0        0  …              0
1104           0              0        0        0  …              0

      35 (1) (7) GDPR  13 (1) e) GDPR  5 (1) a) b) d) e) GDPR  35 (2) GDPR  \
1100                0               0                       0            0
1101                0               0                       0            0
1102                0               0                       0            0
1103                0               0                       0            0
1104                0               0                       0            0
```

```
      22 (2) LSSI  2-ter Codice della privacy   2-octies Codice della privacy  \
1100            0                           0                              0
1101            0                           0                              0
1102            0                           0                              0
1103            0                           0                              0
1104            0                           0                              0

      17 (1) b) GDPR         Id
1100               0  ETid-1143
1101               0  ETid-1144
1102               0  ETid-1145
1103               0  ETid-1146
1104               0  ETid-1147

[5 rows x 245 columns]
```

[8]:
```python
# Categorical Article representation, including other articles and only the␣
 ↪general category of GDPR quoted article
only_article_rep_df = create_art_type_features_df(article_representation_df)
only_article_rep_matrix = categorical_representation_df(only_article_rep_df,2)
ids_names = pd.DataFrame(list(only_article_rep_matrix.index.values))
categorical_gen_article_df = merge_dataset(ids_names, 'Data/
 ↪GDPR_Gen_Article_Matrix.csv')
save_dataset(categorical_gen_article_df, "Categorical_Gen_GDPR_Article_Df")
print('The shape is', categorical_gen_article_df.shape)
# tail of the dataset
categorical_gen_article_df.tail(5)
```

```
The shape is (1105, 63)
```

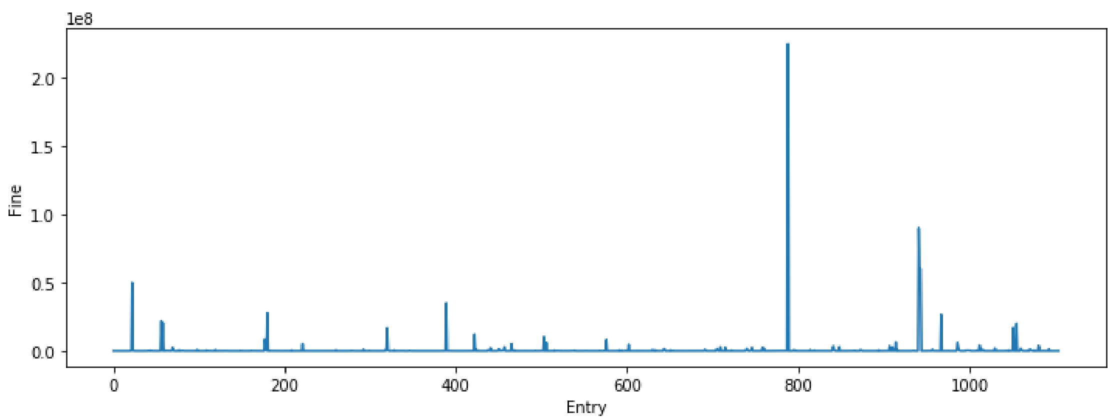
[8]:
```
      GDPR13  GDPR5  GDPR14  GDPR6  GDPR15  GDPR32  GDPR28  GDPR33  GDPR34  \
1100       0      0       0      0       0       0       0       0       0
1101       0      1       0      0       0       0       0       0       0
1102       0      0       0      0       0       0       0       0       0
1103       0      0       0      0       0       0       0       0       0
1104       0      1       0      1       0       0       0       0       0

      GDPR12  …     157 Codice della privacy  GDPR26  GDPR10  \
1100       0  …                            0       0       0
1101       0  …                            0       0       0
1102       0  …                            0       0       0
1103       0  …                            0       0       0
1104       0  …                            0       0       0

      2-ter Codice della privacy  157 Codice della privacy   \
1100                           0                          0
```

```
1101                            0                          0
1102                            0                          0
1103                            0                          0
1104                            0                          0

      166 (2) Codice della privacy  22 (2) LSSI  2-ter Codice della privacy   \
1100                            0            0                           0
1101                            0            0                           0
1102                            0            0                           0
1103                            0            0                           0
1104                            0            0                           0

      2-octies Codice della privacy         Id
1100                             0  ETid-1143
1101                             0  ETid-1144
1102                             0  ETid-1145
1103                             0  ETid-1146
1104                             0  ETid-1147

[5 rows x 63 columns]
```

Target feature: Risk of getting fined (based on the estimated fine to be paid: *Fine* feature)

[9]:
```python
df_Tf_Idf = pd.read_csv('Data/GDPR_Summary_Tf_Idf.csv')
ax = df_Tf_Idf['Fine'].plot(figsize=(12, 4))
ax.set_xlabel("Entry")
ax.set_ylabel("Fine")
```

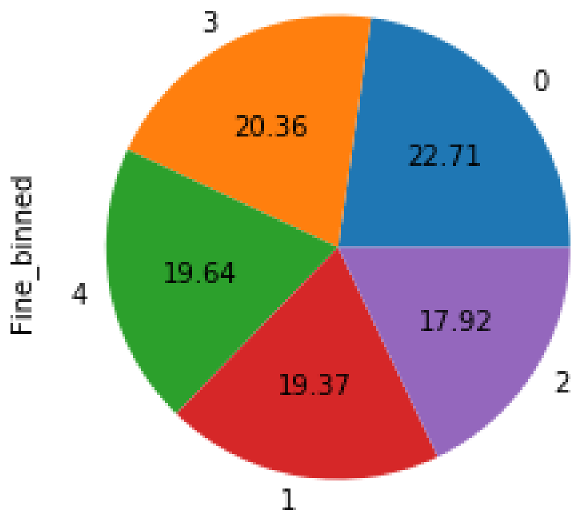
[9]:
```
Text(0, 0.5, 'Fine')
```



[10]:
```python
import seaborn as sns
df_Tf_Idf_new = df_Tf_Idf.iloc[:,:-1]
```

```
df_Tf_Idf_new['Fine_binned'] = pd.qcut(df_Tf_Idf_new['Fine'], 5, labels=False)
df_Tf_Idf_new['Fine_binned'].value_counts().plot(kind="pie", autopct="%0.2f")
```

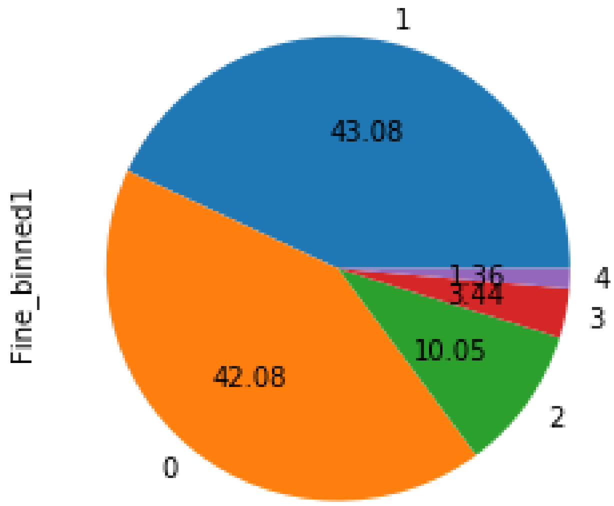
[10]: <matplotlib.axes._subplots.AxesSubplot at 0x25a226d5668>



Binning or discretization is used for the transformation of the *Fine* numerical variable into a categorical feature

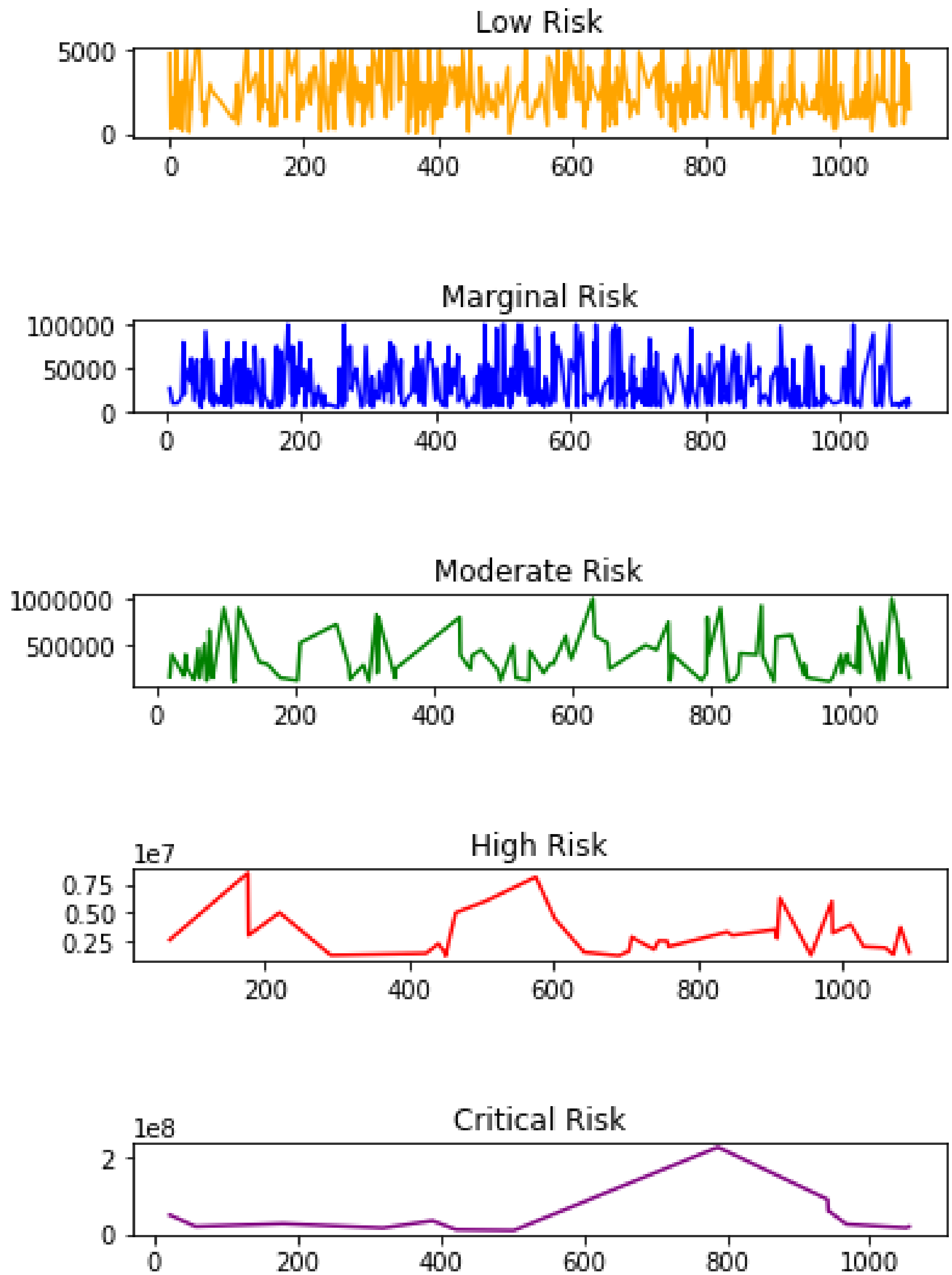
[11]:
```
cut_bins = [0, 5000, 100000, 1000000, 10000000, 500000000]
df_Tf_Idf_new['Fine_binned1'] = pd.cut(df_Tf_Idf_new['Fine'], bins=cut_bins,␣
 ↪labels=False)
df_Tf_Idf_new['Fine_binned1'] = df_Tf_Idf_new['Fine_binned1'].astype(np.int64)
df_Tf_Idf_new['Fine_binned1'].value_counts().plot(kind="pie", autopct="%0.2f")
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x25a1ecd1ba8>



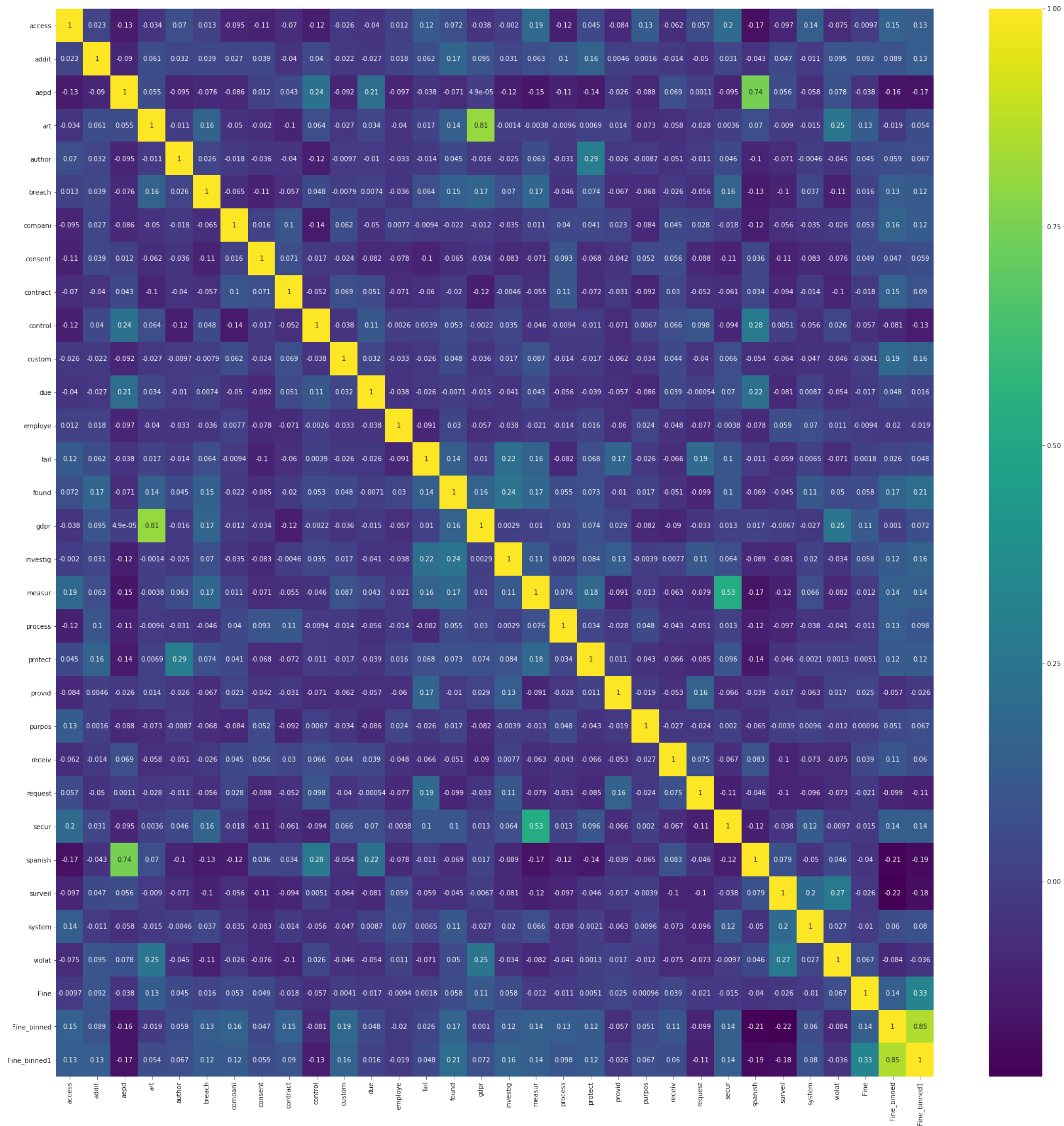
[12]:
```
fig, axes = pyplot.subplots(nrows=5, ncols=1,figsize=(6, 9))
pyplot.subplots_adjust(wspace=2, hspace=2)
df_Tf_Idf_new[df_Tf_Idf_new.Fine_binned1 == 0].Fine.
 ↪plot(ax=axes[0],color="orange")
axes[0].set_title("Low Risk")
df_Tf_Idf_new[df_Tf_Idf_new.Fine_binned1 == 1].Fine.
 ↪plot(ax=axes[1],color="blue")
axes[1].set_title("Marginal Risk")
df_Tf_Idf_new[df_Tf_Idf_new.Fine_binned1 == 2].Fine.
 ↪plot(ax=axes[2],color="green")
axes[2].set_title("Moderate Risk")
df_Tf_Idf_new[df_Tf_Idf_new.Fine_binned1 == 3].Fine.plot(ax=axes[3],color="red")
axes[3].set_title("High Risk")
df_Tf_Idf_new[df_Tf_Idf_new.Fine_binned1 == 4].Fine.
 ↪plot(ax=axes[4],color="purple")
axes[4].set_title("Critical Risk")
```

[12]:
```
Text(0.5, 1.0, 'Critical Risk')
```



[13]:
```
# The heatmap is produced according to the feature correlations
pyplot.figure(figsize=(30,30))
sns.heatmap(df_Tf_Idf_new.corr(),annot=True,cmap='viridis')
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x25a226a4e10>



6 Selection/development of the prediction models

Based on the defined goal, we have to evaluate the combinations of modeling representation techniques by using classical machine learning approaches

[14]:
```
# Training and evaluation
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
```

[15]:

```python
# List of models to compare
seed = 7
models = []
models.append(('LR',
 ↪LogisticRegression(random_state=0,solver='lbfgs',multi_class='auto')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', MultinomialNB()))
models.append(('SVM', LinearSVC()))
models.append(('RANFOR',RandomForestClassifier(n_estimators=200, max_depth=3,
 ↪random_state=0)))
```

[16]:

```python
X = df_Tf_Idf_new.iloc[:,0:len(df_Tf_Idf_new.columns)-3]
Y = df_Tf_Idf_new['Fine_binned1']
```

[17]:

```python
# Evaluate each model in turn
results = []
names = []
scoring = 'Accuracy'
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold)
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
```

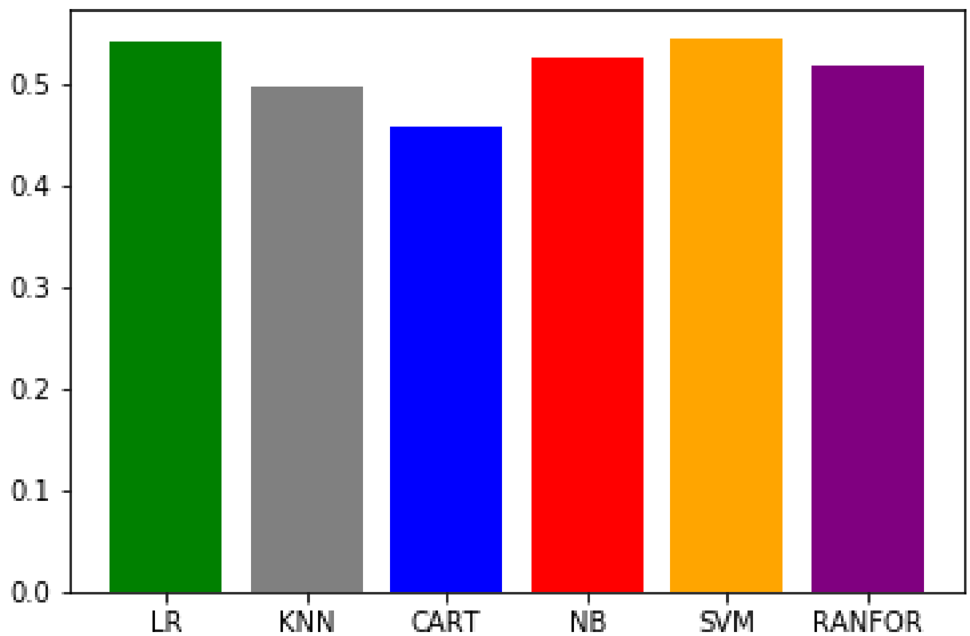
```
LR: 0.542154 (0.055581)
KNN: 0.496970 (0.055986)
CART: 0.457985 (0.048227)
NB: 0.525864 (0.045428)
SVM: 0.544840 (0.051505)
RANFOR: 0.518698 (0.052974)
```

[18]:

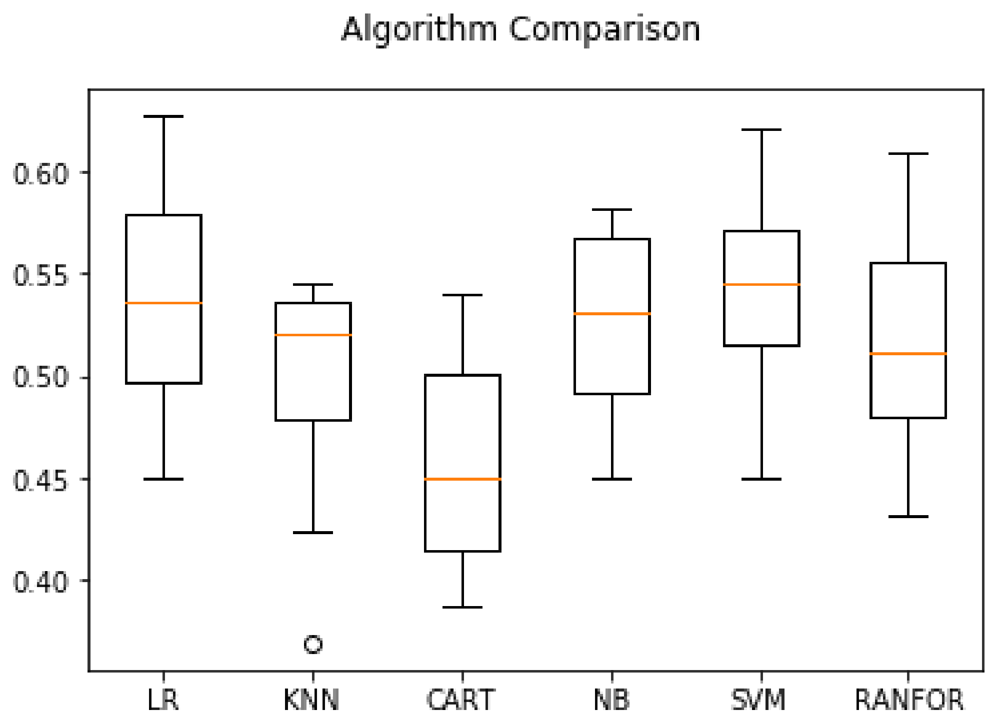
```python
results_mean = [results[i].mean() for i in range(len(results))]
colors = ['green', 'grey', 'blue', 'red', 'orange', 'purple']
```

```
pyplot.bar(names, results_mean,color=colors)
```

[18]: <BarContainer object of 6 artists>



[19]:
```
# boxplot algorithm comparison
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```



7 Building deep model

[20]:
```
import tensorflow as tf
from tensorflow import keras
from keras.layers import Dense, Dropout
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from keras.constraints import unit_norm
from keras.regularizers import l2

model = Sequential()

model.add(Dense(units=32, activation='relu', input_dim=(29),␣
 ↪kernel_constraint=unit_norm()))

model.add(Dense(units=24, activation='relu'))
model.add(Dropout(.2))

model.add(Dense(units=11, activation='relu'))
model.add(Dropout(.2))

model.add(Dense(units=8, activation='relu'))
```

```
model.add(Dense(units=5, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
↪metrics=['accuracy'])
model.summary()
```

```
Using TensorFlow backend.

Model: "sequential_1"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                960
_________________________________________________________________
dense_2 (Dense)              (None, 24)                792
_________________________________________________________________
dropout_1 (Dropout)          (None, 24)                0
_________________________________________________________________
dense_3 (Dense)              (None, 11)                275
_________________________________________________________________
dropout_2 (Dropout)          (None, 11)                0
_________________________________________________________________
dense_4 (Dense)              (None, 8)                 96
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 45
=================================================================
Total params: 2,168
Trainable params: 2,168
Non-trainable params: 0

_________________________________________________________________
```

[21]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33,␣
↪random_state=42)
history = model.fit(X_train, Y_train, epochs=100,␣
↪batch_size=32,validation_data=(X_test, Y_test))
```

```
Train on 740 samples, validate on 365 samples
Epoch 1/100
740/740 [==============================] - 1s 1ms/step - loss: 1.5881 -
accuracy: 0.3878 - val_loss: 1.5633 - val_accuracy: 0.4932
Epoch 2/100
740/740 [==============================] - 0s 81us/step - loss: 1.5400 -
accuracy: 0.4284 - val_loss: 1.5012 - val_accuracy: 0.4795
Epoch 3/100
740/740 [==============================] - 0s 81us/step - loss: 1.4665 -
accuracy: 0.4541 - val_loss: 1.4084 - val_accuracy: 0.4849
Epoch 4/100
```

```
740/740 [==============================] - 0s 78us/step - loss: 1.3607 -
accuracy: 0.4527 - val_loss: 1.2914 - val_accuracy: 0.4795
Epoch 5/100
740/740 [==============================] - ETA: 0s - loss: 1.4547 - accuracy:
0.28 - 0s 80us/step - loss: 1.2859 - accuracy: 0.4662 - val_loss: 1.2176 -
val_accuracy: 0.4959
Epoch 6/100
740/740 [==============================] - 0s 80us/step - loss: 1.1852 -
accuracy: 0.5297 - val_loss: 1.1636 - val_accuracy: 0.5260
Epoch 7/100
740/740 [==============================] - 0s 78us/step - loss: 1.1407 -
accuracy: 0.5311 - val_loss: 1.1339 - val_accuracy: 0.5370
Epoch 8/100
740/740 [==============================] - 0s 85us/step - loss: 1.0904 -
accuracy: 0.5554 - val_loss: 1.1183 - val_accuracy: 0.5370
Epoch 9/100
740/740 [==============================] - 0s 90us/step - loss: 1.0775 -
accuracy: 0.5473 - val_loss: 1.1057 - val_accuracy: 0.5288
Epoch 10/100
740/740 [==============================] - 0s 81us/step - loss: 1.0814 -
accuracy: 0.5662 - val_loss: 1.0910 - val_accuracy: 0.5260
Epoch 11/100
740/740 [==============================] - 0s 90us/step - loss: 1.0644 -
accuracy: 0.5635 - val_loss: 1.0846 - val_accuracy: 0.5260
Epoch 12/100
740/740 [==============================] - 0s 86us/step - loss: 1.0317 -
accuracy: 0.5568 - val_loss: 1.0827 - val_accuracy: 0.5178
Epoch 13/100
740/740 [==============================] - 0s 85us/step - loss: 1.0370 -
accuracy: 0.5649 - val_loss: 1.0709 - val_accuracy: 0.5178
Epoch 14/100
740/740 [==============================] - 0s 93us/step - loss: 1.0321 -
accuracy: 0.5662 - val_loss: 1.0669 - val_accuracy: 0.5205
Epoch 15/100
740/740 [==============================] - 0s 78us/step - loss: 1.0048 -
accuracy: 0.5784 - val_loss: 1.0628 - val_accuracy: 0.5288
Epoch 16/100
740/740 [==============================] - 0s 86us/step - loss: 1.0065 -
accuracy: 0.5649 - val_loss: 1.0568 - val_accuracy: 0.5315
Epoch 17/100
740/740 [==============================] - 0s 90us/step - loss: 0.9962 -
accuracy: 0.5851 - val_loss: 1.0569 - val_accuracy: 0.5205
Epoch 18/100
740/740 [==============================] - 0s 88us/step - loss: 0.9918 -
accuracy: 0.5986 - val_loss: 1.0535 - val_accuracy: 0.5288
Epoch 19/100
740/740 [==============================] - 0s 88us/step - loss: 1.0107 -
accuracy: 0.5662 - val_loss: 1.0509 - val_accuracy: 0.5233
```

```
Epoch 20/100
740/740 [==============================] - 0s 84us/step - loss: 0.9787 -
accuracy: 0.5959 - val_loss: 1.0373 - val_accuracy: 0.5397
Epoch 21/100
740/740 [==============================] - 0s 84us/step - loss: 0.9716 -
accuracy: 0.5973 - val_loss: 1.0418 - val_accuracy: 0.5315
Epoch 22/100
740/740 [==============================] - 0s 97us/step - loss: 0.9679 -
accuracy: 0.5905 - val_loss: 1.0344 - val_accuracy: 0.5425
Epoch 23/100
740/740 [==============================] - 0s 93us/step - loss: 0.9826 -
accuracy: 0.5851 - val_loss: 1.0376 - val_accuracy: 0.5315
Epoch 24/100
740/740 [==============================] - 0s 90us/step - loss: 0.9510 -
accuracy: 0.5838 - val_loss: 1.0384 - val_accuracy: 0.5288
Epoch 25/100
740/740 [==============================] - 0s 90us/step - loss: 0.9556 -
accuracy: 0.5784 - val_loss: 1.0394 - val_accuracy: 0.5342
Epoch 26/100
740/740 [==============================] - 0s 84us/step - loss: 0.9600 -
accuracy: 0.5824 - val_loss: 1.0378 - val_accuracy: 0.5342
Epoch 27/100
740/740 [==============================] - 0s 140us/step - loss: 0.9568 -
accuracy: 0.6041 - val_loss: 1.0364 - val_accuracy: 0.5288
Epoch 28/100
740/740 [==============================] - 0s 78us/step - loss: 0.9428 -
accuracy: 0.6122 - val_loss: 1.0375 - val_accuracy: 0.5205
Epoch 29/100
740/740 [==============================] - 0s 86us/step - loss: 0.9323 -
accuracy: 0.6041 - val_loss: 1.0374 - val_accuracy: 0.5342
Epoch 30/100
740/740 [==============================] - 0s 85us/step - loss: 0.9522 -
accuracy: 0.5892 - val_loss: 1.0357 - val_accuracy: 0.5151
Epoch 31/100
740/740 [==============================] - 0s 93us/step - loss: 0.9347 -
accuracy: 0.6176 - val_loss: 1.0330 - val_accuracy: 0.5041
Epoch 32/100
740/740 [==============================] - 0s 89us/step - loss: 0.9345 -
accuracy: 0.6014 - val_loss: 1.0320 - val_accuracy: 0.5151
Epoch 33/100
740/740 [==============================] - 0s 86us/step - loss: 0.9228 -
accuracy: 0.6000 - val_loss: 1.0329 - val_accuracy: 0.5123
Epoch 34/100
740/740 [==============================] - 0s 85us/step - loss: 0.9170 -
accuracy: 0.6122 - val_loss: 1.0257 - val_accuracy: 0.5205
Epoch 35/100
740/740 [==============================] - 0s 90us/step - loss: 0.9214 -
accuracy: 0.6041 - val_loss: 1.0321 - val_accuracy: 0.5178
```

```
Epoch 36/100
740/740 [==============================] - 0s 89us/step - loss: 0.9140 -
accuracy: 0.5905 - val_loss: 1.0318 - val_accuracy: 0.5041
Epoch 37/100
740/740 [==============================] - 0s 85us/step - loss: 0.9031 -
accuracy: 0.6108 - val_loss: 1.0292 - val_accuracy: 0.5096
Epoch 38/100
740/740 [==============================] - 0s 85us/step - loss: 0.9076 -
accuracy: 0.6095 - val_loss: 1.0304 - val_accuracy: 0.5151
Epoch 39/100
740/740 [==============================] - 0s 94us/step - loss: 0.9018 -
accuracy: 0.6243 - val_loss: 1.0361 - val_accuracy: 0.5178
Epoch 40/100
740/740 [==============================] - 0s 90us/step - loss: 0.9001 -
accuracy: 0.6041 - val_loss: 1.0369 - val_accuracy: 0.5151
Epoch 41/100
740/740 [==============================] - 0s 86us/step - loss: 0.8954 -
accuracy: 0.6149 - val_loss: 1.0340 - val_accuracy: 0.5151
Epoch 42/100
740/740 [==============================] - 0s 75us/step - loss: 0.8737 -
accuracy: 0.6378 - val_loss: 1.0459 - val_accuracy: 0.5041
Epoch 43/100
740/740 [==============================] - 0s 77us/step - loss: 0.8746 -
accuracy: 0.6230 - val_loss: 1.0381 - val_accuracy: 0.5315
Epoch 44/100
740/740 [==============================] - 0s 73us/step - loss: 0.8820 -
accuracy: 0.6311 - val_loss: 1.0390 - val_accuracy: 0.5260
Epoch 45/100
740/740 [==============================] - 0s 71us/step - loss: 0.8782 -
accuracy: 0.6230 - val_loss: 1.0424 - val_accuracy: 0.5233
Epoch 46/100
740/740 [==============================] - 0s 77us/step - loss: 0.8606 -
accuracy: 0.6473 - val_loss: 1.0484 - val_accuracy: 0.5205
Epoch 47/100
740/740 [==============================] - 0s 85us/step - loss: 0.8505 -
accuracy: 0.6338 - val_loss: 1.0483 - val_accuracy: 0.5260
Epoch 48/100
740/740 [==============================] - 0s 90us/step - loss: 0.8561 -
accuracy: 0.6446 - val_loss: 1.0630 - val_accuracy: 0.5233
Epoch 49/100
740/740 [==============================] - 0s 106us/step - loss: 0.8679 -
accuracy: 0.6189 - val_loss: 1.0533 - val_accuracy: 0.5288
Epoch 50/100
740/740 [==============================] - 0s 84us/step - loss: 0.8600 -
accuracy: 0.6392 - val_loss: 1.0552 - val_accuracy: 0.5260
Epoch 51/100
740/740 [==============================] - 0s 74us/step - loss: 0.8680 -
accuracy: 0.6324 - val_loss: 1.0377 - val_accuracy: 0.5397
```

```
Epoch 52/100
740/740 [==============================] - 0s 75us/step - loss: 0.8390 -
accuracy: 0.6459 - val_loss: 1.0545 - val_accuracy: 0.5260
Epoch 53/100
740/740 [==============================] - 0s 82us/step - loss: 0.8378 -
accuracy: 0.6446 - val_loss: 1.0555 - val_accuracy: 0.5315
Epoch 54/100
740/740 [==============================] - 0s 75us/step - loss: 0.8460 -
accuracy: 0.6432 - val_loss: 1.0625 - val_accuracy: 0.5288
Epoch 55/100
740/740 [==============================] - 0s 80us/step - loss: 0.8400 -
accuracy: 0.6541 - val_loss: 1.0646 - val_accuracy: 0.5260
Epoch 56/100
740/740 [==============================] - 0s 71us/step - loss: 0.8387 -
accuracy: 0.6541 - val_loss: 1.0613 - val_accuracy: 0.5288
Epoch 57/100
740/740 [==============================] - 0s 75us/step - loss: 0.8145 -
accuracy: 0.6527 - val_loss: 1.0680 - val_accuracy: 0.5315
Epoch 58/100
740/740 [==============================] - 0s 71us/step - loss: 0.8203 -
accuracy: 0.6527 - val_loss: 1.0711 - val_accuracy: 0.5342
Epoch 59/100
740/740 [==============================] - 0s 71us/step - loss: 0.8138 -
accuracy: 0.6473 - val_loss: 1.0727 - val_accuracy: 0.5288
Epoch 60/100
740/740 [==============================] - 0s 71us/step - loss: 0.8056 -
accuracy: 0.6716 - val_loss: 1.0926 - val_accuracy: 0.5288
Epoch 61/100
740/740 [==============================] - 0s 82us/step - loss: 0.8258 -
accuracy: 0.6459 - val_loss: 1.0831 - val_accuracy: 0.5123
Epoch 62/100
740/740 [==============================] - 0s 90us/step - loss: 0.7936 -
accuracy: 0.6703 - val_loss: 1.0785 - val_accuracy: 0.5315
Epoch 63/100
740/740 [==============================] - 0s 94us/step - loss: 0.7913 -
accuracy: 0.6703 - val_loss: 1.0841 - val_accuracy: 0.5288
Epoch 64/100
740/740 [==============================] - 0s 77us/step - loss: 0.7830 -
accuracy: 0.6622 - val_loss: 1.1016 - val_accuracy: 0.5452
Epoch 65/100
740/740 [==============================] - 0s 73us/step - loss: 0.7837 -
accuracy: 0.6635 - val_loss: 1.0901 - val_accuracy: 0.5425
Epoch 66/100
740/740 [==============================] - 0s 73us/step - loss: 0.7604 -
accuracy: 0.6919 - val_loss: 1.0877 - val_accuracy: 0.5233
Epoch 67/100
740/740 [==============================] - 0s 82us/step - loss: 0.7808 -
accuracy: 0.6878 - val_loss: 1.0989 - val_accuracy: 0.5260
```

```
Epoch 68/100
740/740 [==============================] - 0s 94us/step - loss: 0.7947 -
accuracy: 0.6581 - val_loss: 1.0760 - val_accuracy: 0.5260
Epoch 69/100
740/740 [==============================] - 0s 90us/step - loss: 0.7802 -
accuracy: 0.6770 - val_loss: 1.0962 - val_accuracy: 0.5233
Epoch 70/100
740/740 [==============================] - 0s 80us/step - loss: 0.7570 -
accuracy: 0.6986 - val_loss: 1.0950 - val_accuracy: 0.5288
Epoch 71/100
740/740 [==============================] - 0s 73us/step - loss: 0.7642 -
accuracy: 0.6878 - val_loss: 1.1190 - val_accuracy: 0.5452
Epoch 72/100
740/740 [==============================] - 0s 78us/step - loss: 0.7491 -
accuracy: 0.6838 - val_loss: 1.1201 - val_accuracy: 0.5151
Epoch 73/100
740/740 [==============================] - 0s 74us/step - loss: 0.7487 -
accuracy: 0.7041 - val_loss: 1.1293 - val_accuracy: 0.5397
Epoch 74/100
740/740 [==============================] - 0s 78us/step - loss: 0.7312 -
accuracy: 0.7054 - val_loss: 1.1227 - val_accuracy: 0.5288
Epoch 75/100
740/740 [==============================] - 0s 71us/step - loss: 0.7821 -
accuracy: 0.6851 - val_loss: 1.1200 - val_accuracy: 0.5260
Epoch 76/100
740/740 [==============================] - 0s 78us/step - loss: 0.7589 -
accuracy: 0.6689 - val_loss: 1.1252 - val_accuracy: 0.5397
Epoch 77/100
740/740 [==============================] - 0s 73us/step - loss: 0.7205 -
accuracy: 0.7149 - val_loss: 1.1236 - val_accuracy: 0.5288
Epoch 78/100
740/740 [==============================] - 0s 78us/step - loss: 0.7360 -
accuracy: 0.6757 - val_loss: 1.1257 - val_accuracy: 0.5370
Epoch 79/100
740/740 [==============================] - 0s 73us/step - loss: 0.7146 -
accuracy: 0.7014 - val_loss: 1.1470 - val_accuracy: 0.5370
Epoch 80/100
740/740 [==============================] - 0s 78us/step - loss: 0.7214 -
accuracy: 0.6919 - val_loss: 1.1538 - val_accuracy: 0.5452
Epoch 81/100
740/740 [==============================] - 0s 74us/step - loss: 0.7250 -
accuracy: 0.7068 - val_loss: 1.1593 - val_accuracy: 0.5260
Epoch 82/100
740/740 [==============================] - 0s 80us/step - loss: 0.6866 -
accuracy: 0.7189 - val_loss: 1.1709 - val_accuracy: 0.5507
Epoch 83/100
740/740 [==============================] - 0s 74us/step - loss: 0.7081 -
accuracy: 0.7068 - val_loss: 1.1968 - val_accuracy: 0.5370
```

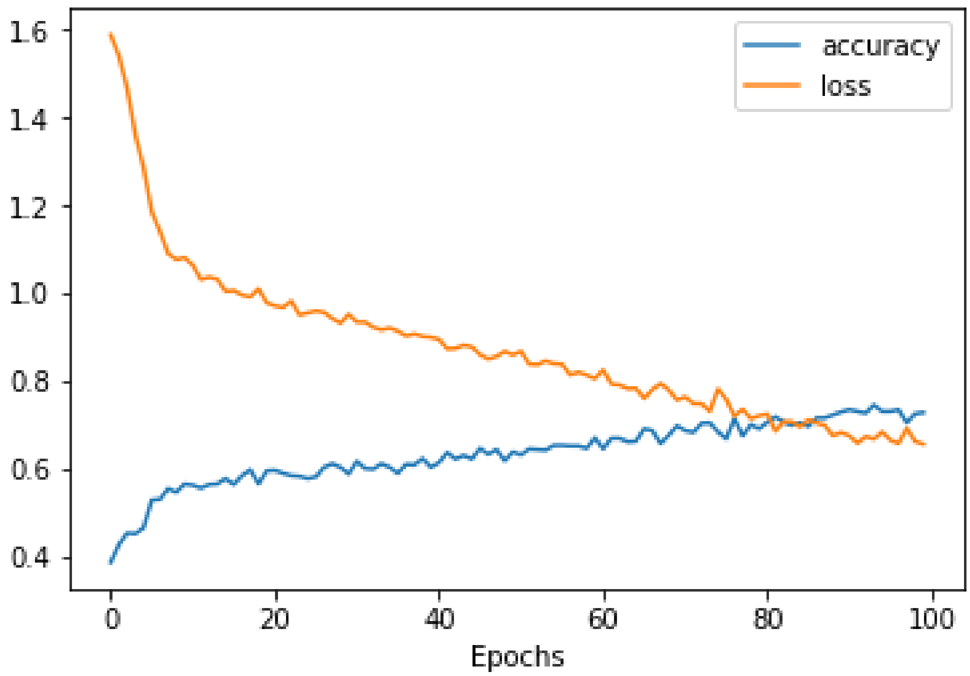
```
Epoch 84/100
740/740 [==============================] - ETA: 0s - loss: 0.7977 - accuracy:
0.71 - 0s 82us/step - loss: 0.7075 - accuracy: 0.7000 - val_loss: 1.1771 -
val_accuracy: 0.5233
Epoch 85/100
740/740 [==============================] - 0s 84us/step - loss: 0.6963 -
accuracy: 0.7054 - val_loss: 1.1809 - val_accuracy: 0.5315
Epoch 86/100
740/740 [==============================] - 0s 80us/step - loss: 0.7122 -
accuracy: 0.6973 - val_loss: 1.1809 - val_accuracy: 0.5233
Epoch 87/100
740/740 [==============================] - 0s 77us/step - loss: 0.7052 -
accuracy: 0.7162 - val_loss: 1.1737 - val_accuracy: 0.5233
Epoch 88/100
740/740 [==============================] - 0s 88us/step - loss: 0.6999 -
accuracy: 0.7162 - val_loss: 1.1764 - val_accuracy: 0.5260
Epoch 89/100
740/740 [==============================] - 0s 84us/step - loss: 0.6760 -
accuracy: 0.7230 - val_loss: 1.1978 - val_accuracy: 0.5370
Epoch 90/100
740/740 [==============================] - 0s 75us/step - loss: 0.6835 -
accuracy: 0.7297 - val_loss: 1.2078 - val_accuracy: 0.5397
Epoch 91/100
740/740 [==============================] - 0s 78us/step - loss: 0.6760 -
accuracy: 0.7351 - val_loss: 1.2303 - val_accuracy: 0.5397
Epoch 92/100
740/740 [==============================] - 0s 75us/step - loss: 0.6591 -
accuracy: 0.7311 - val_loss: 1.2243 - val_accuracy: 0.5452
Epoch 93/100
740/740 [==============================] - 0s 82us/step - loss: 0.6744 -
accuracy: 0.7270 - val_loss: 1.2157 - val_accuracy: 0.5315
Epoch 94/100
740/740 [==============================] - 0s 90us/step - loss: 0.6685 -
accuracy: 0.7459 - val_loss: 1.2346 - val_accuracy: 0.5260
Epoch 95/100
740/740 [==============================] - 0s 86us/step - loss: 0.6847 -
accuracy: 0.7311 - val_loss: 1.2241 - val_accuracy: 0.5397
Epoch 96/100
740/740 [==============================] - 0s 85us/step - loss: 0.6663 -
accuracy: 0.7311 - val_loss: 1.2198 - val_accuracy: 0.5315
Epoch 97/100
740/740 [==============================] - 0s 88us/step - loss: 0.6583 -
accuracy: 0.7351 - val_loss: 1.2184 - val_accuracy: 0.5288
Epoch 98/100
740/740 [==============================] - 0s 80us/step - loss: 0.6930 -
accuracy: 0.7054 - val_loss: 1.2088 - val_accuracy: 0.5178
Epoch 99/100
740/740 [==============================] - 0s 78us/step - loss: 0.6633 -
```

```
accuracy: 0.7257 - val_loss: 1.2367 - val_accuracy: 0.5370
Epoch 100/100
740/740 [==============================] - 0s 77us/step - loss: 0.6569 -
accuracy: 0.7284 - val_loss: 1.2599 - val_accuracy: 0.5288
```

[22]:
```
val_acc_test = []

val_loss_test1, val_acc_test1 = model.evaluate(X_test, Y_test)
val_acc_test.append(val_acc_test1)

print('Test accuracy:', val_acc_test1)
print('Test loss:', val_loss_test1)

val_loss_train1, val_acc_train1 = model.evaluate(X_train, Y_train)

print('Train accuracy:', val_acc_train1)
print('Train loss:', val_loss_train1)
```

```
365/365 [==============================] - 0s 44us/step
Test accuracy: 0.5287671089172363
Test loss: 1.2599353999307712
740/740 [==============================] - 0s 34us/step
Train accuracy: 0.7662162184715271
Train loss: 0.5784212410449981
```

[23]:
```
pyplot.plot(history.history['accuracy'])
pyplot.plot(history.history['loss'])
pyplot.xlabel("Epochs")
pyplot.legend(['accuracy','loss'])
pyplot.show()
```



Reducing the Network's Capacity (RNC)

[24]:
```
model_RNC = Sequential()

model_RNC.add(Dense(units=25, activation='relu', input_dim=(29),␣
 ↪kernel_constraint=unit_norm()))

model_RNC.add(Dense(units=15, activation='relu'))
model_RNC.add(Dropout(.2))

model_RNC.add(Dense(units=5, activation='softmax'))

model_RNC.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
model_RNC.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 25)                750
_________________________________________________________________
dense_7 (Dense)              (None, 15)                390
_________________________________________________________________
dropout_3 (Dropout)          (None, 15)                0
```

```
_________________________________________________________________
dense_8 (Dense)              (None, 5)                 80
=================================================================
Total params: 1,220
Trainable params: 1,220
Non-trainable params: 0
_________________________________________________________________
```

[30]:
```python
history_RNC=model_RNC.fit(X_train, Y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
740/740 [==============================] - 0s 54us/step - loss: 0.6878 -
accuracy: 0.7311
Epoch 2/100
740/740 [==============================] - 0s 49us/step - loss: 0.6943 -
accuracy: 0.7230
Epoch 3/100
740/740 [==============================] - 0s 59us/step - loss: 0.6837 -
accuracy: 0.7189
Epoch 4/100
740/740 [==============================] - 0s 53us/step - loss: 0.6984 -
accuracy: 0.7122
Epoch 5/100
740/740 [==============================] - 0s 57us/step - loss: 0.7082 -
accuracy: 0.7122
Epoch 6/100
740/740 [==============================] - 0s 54us/step - loss: 0.6890 -
accuracy: 0.7284
Epoch 7/100
740/740 [==============================] - 0s 57us/step - loss: 0.6947 -
accuracy: 0.7162
Epoch 8/100
740/740 [==============================] - 0s 54us/step - loss: 0.6871 -
accuracy: 0.7162
Epoch 9/100
740/740 [==============================] - 0s 59us/step - loss: 0.7003 -
accuracy: 0.7081
Epoch 10/100
740/740 [==============================] - 0s 57us/step - loss: 0.6969 -
accuracy: 0.7149
Epoch 11/100
740/740 [==============================] - 0s 50us/step - loss: 0.6793 -
accuracy: 0.7365
Epoch 12/100
740/740 [==============================] - 0s 61us/step - loss: 0.6783 -
accuracy: 0.7270
Epoch 13/100
740/740 [==============================] - 0s 55us/step - loss: 0.6816 -
```

```
accuracy: 0.7189
Epoch 14/100
740/740 [==============================] - 0s 55us/step - loss: 0.6810 -
accuracy: 0.7203
Epoch 15/100
740/740 [==============================] - 0s 61us/step - loss: 0.6853 -
accuracy: 0.7162
Epoch 16/100
740/740 [==============================] - 0s 62us/step - loss: 0.6993 -
accuracy: 0.7162
Epoch 17/100
740/740 [==============================] - 0s 58us/step - loss: 0.6822 -
accuracy: 0.7351
Epoch 18/100
740/740 [==============================] - 0s 51us/step - loss: 0.6682 -
accuracy: 0.7338
Epoch 19/100
740/740 [==============================] - 0s 61us/step - loss: 0.7045 -
accuracy: 0.7108
Epoch 20/100
740/740 [==============================] - 0s 61us/step - loss: 0.6669 -
accuracy: 0.7419
Epoch 21/100
740/740 [==============================] - 0s 54us/step - loss: 0.6903 -
accuracy: 0.7176
Epoch 22/100
740/740 [==============================] - 0s 57us/step - loss: 0.6755 -
accuracy: 0.7378
Epoch 23/100
740/740 [==============================] - 0s 54us/step - loss: 0.6832 -
accuracy: 0.7041
Epoch 24/100
740/740 [==============================] - 0s 59us/step - loss: 0.6774 -
accuracy: 0.7284
Epoch 25/100
740/740 [==============================] - 0s 73us/step - loss: 0.6753 -
accuracy: 0.7189
Epoch 26/100
740/740 [==============================] - 0s 59us/step - loss: 0.6776 -
accuracy: 0.7257
Epoch 27/100
740/740 [==============================] - 0s 73us/step - loss: 0.6725 -
accuracy: 0.7324
Epoch 28/100
740/740 [==============================] - 0s 61us/step - loss: 0.6593 -
accuracy: 0.7378
Epoch 29/100
740/740 [==============================] - 0s 58us/step - loss: 0.6588 -
```

```
accuracy: 0.7378
Epoch 30/100
740/740 [==============================] - 0s 58us/step - loss: 0.6633 -
accuracy: 0.7446
Epoch 31/100
740/740 [==============================] - 0s 54us/step - loss: 0.6697 -
accuracy: 0.7257
Epoch 32/100
740/740 [==============================] - 0s 50us/step - loss: 0.6848 -
accuracy: 0.7162
Epoch 33/100
740/740 [==============================] - 0s 65us/step - loss: 0.6536 -
accuracy: 0.7459
Epoch 34/100
740/740 [==============================] - 0s 70us/step - loss: 0.6589 -
accuracy: 0.7568
Epoch 35/100
740/740 [==============================] - 0s 59us/step - loss: 0.6734 -
accuracy: 0.7216
Epoch 36/100
740/740 [==============================] - 0s 57us/step - loss: 0.6700 -
accuracy: 0.7473
Epoch 37/100
740/740 [==============================] - 0s 63us/step - loss: 0.6585 -
accuracy: 0.7351
Epoch 38/100
740/740 [==============================] - 0s 58us/step - loss: 0.6641 -
accuracy: 0.7257
Epoch 39/100
740/740 [==============================] - 0s 54us/step - loss: 0.6567 -
accuracy: 0.7365
Epoch 40/100
740/740 [==============================] - 0s 58us/step - loss: 0.6612 -
accuracy: 0.7338
Epoch 41/100
740/740 [==============================] - 0s 59us/step - loss: 0.6335 -
accuracy: 0.7527
Epoch 42/100
740/740 [==============================] - 0s 66us/step - loss: 0.6671 -
accuracy: 0.7270
Epoch 43/100
740/740 [==============================] - 0s 66us/step - loss: 0.6545 -
accuracy: 0.7419
Epoch 44/100
740/740 [==============================] - 0s 63us/step - loss: 0.6524 -
accuracy: 0.7432
Epoch 45/100
740/740 [==============================] - 0s 57us/step - loss: 0.6688 -
```

```
accuracy: 0.7365
Epoch 46/100
740/740 [==============================] - 0s 54us/step - loss: 0.6566 -
accuracy: 0.7514
Epoch 47/100
740/740 [==============================] - 0s 54us/step - loss: 0.6458 -
accuracy: 0.7473
Epoch 48/100
740/740 [==============================] - 0s 57us/step - loss: 0.6522 -
accuracy: 0.7351
Epoch 49/100
740/740 [==============================] - 0s 54us/step - loss: 0.6411 -
accuracy: 0.7554
Epoch 50/100
740/740 [==============================] - 0s 53us/step - loss: 0.6580 -
accuracy: 0.7554
Epoch 51/100
740/740 [==============================] - 0s 54us/step - loss: 0.6344 -
accuracy: 0.7459
Epoch 52/100
740/740 [==============================] - 0s 50us/step - loss: 0.6294 -
accuracy: 0.7541
Epoch 53/100
740/740 [==============================] - 0s 61us/step - loss: 0.6572 -
accuracy: 0.7257
Epoch 54/100
740/740 [==============================] - 0s 59us/step - loss: 0.6551 -
accuracy: 0.7365
Epoch 55/100
740/740 [==============================] - 0s 59us/step - loss: 0.6522 -
accuracy: 0.7378
Epoch 56/100
740/740 [==============================] - 0s 61us/step - loss: 0.6383 -
accuracy: 0.7324
Epoch 57/100
740/740 [==============================] - 0s 61us/step - loss: 0.6505 -
accuracy: 0.7554
Epoch 58/100
740/740 [==============================] - 0s 58us/step - loss: 0.6254 -
accuracy: 0.7568
Epoch 59/100
740/740 [==============================] - 0s 57us/step - loss: 0.6396 -
accuracy: 0.7459
Epoch 60/100
740/740 [==============================] - 0s 59us/step - loss: 0.6318 -
accuracy: 0.7514
Epoch 61/100
740/740 [==============================] - 0s 50us/step - loss: 0.6263 -
```

```
accuracy: 0.7527
Epoch 62/100
740/740 [==============================] - 0s 47us/step - loss: 0.6423 -
accuracy: 0.7581
Epoch 63/100
740/740 [==============================] - 0s 49us/step - loss: 0.6312 -
accuracy: 0.7419
Epoch 64/100
740/740 [==============================] - 0s 51us/step - loss: 0.6288 -
accuracy: 0.7486
Epoch 65/100
740/740 [==============================] - 0s 50us/step - loss: 0.6371 -
accuracy: 0.7554
Epoch 66/100
740/740 [==============================] - 0s 51us/step - loss: 0.6212 -
accuracy: 0.7635
Epoch 67/100
740/740 [==============================] - 0s 54us/step - loss: 0.6195 -
accuracy: 0.7703
Epoch 68/100
740/740 [==============================] - 0s 50us/step - loss: 0.6279 -
accuracy: 0.7527
Epoch 69/100
740/740 [==============================] - 0s 49us/step - loss: 0.6221 -
accuracy: 0.7527
Epoch 70/100
740/740 [==============================] - 0s 59us/step - loss: 0.6257 -
accuracy: 0.7554
Epoch 71/100
740/740 [==============================] - 0s 58us/step - loss: 0.6504 -
accuracy: 0.7473
Epoch 72/100
740/740 [==============================] - 0s 55us/step - loss: 0.6190 -
accuracy: 0.7581
Epoch 73/100
740/740 [==============================] - 0s 58us/step - loss: 0.6223 -
accuracy: 0.7541
Epoch 74/100
740/740 [==============================] - 0s 62us/step - loss: 0.6340 -
accuracy: 0.7500
Epoch 75/100
740/740 [==============================] - 0s 58us/step - loss: 0.6229 -
accuracy: 0.7581
Epoch 76/100
740/740 [==============================] - 0s 50us/step - loss: 0.6348 -
accuracy: 0.7541
Epoch 77/100
740/740 [==============================] - 0s 53us/step - loss: 0.6216 -
```

```
accuracy: 0.7716
Epoch 78/100
740/740 [==============================] - 0s 51us/step - loss: 0.6330 -
accuracy: 0.7446
Epoch 79/100
740/740 [==============================] - 0s 55us/step - loss: 0.6134 -
accuracy: 0.7689
Epoch 80/100
740/740 [==============================] - 0s 51us/step - loss: 0.6170 -
accuracy: 0.7676
Epoch 81/100
740/740 [==============================] - 0s 53us/step - loss: 0.6182 -
accuracy: 0.7568
Epoch 82/100
740/740 [==============================] - 0s 63us/step - loss: 0.6297 -
accuracy: 0.7527
Epoch 83/100
740/740 [==============================] - 0s 51us/step - loss: 0.6138 -
accuracy: 0.7541
Epoch 84/100
740/740 [==============================] - 0s 50us/step - loss: 0.6218 -
accuracy: 0.7446
Epoch 85/100
740/740 [==============================] - 0s 49us/step - loss: 0.5950 -
accuracy: 0.7703
Epoch 86/100
740/740 [==============================] - 0s 49us/step - loss: 0.6187 -
accuracy: 0.7689
Epoch 87/100
740/740 [==============================] - 0s 50us/step - loss: 0.6064 -
accuracy: 0.7689
Epoch 88/100
740/740 [==============================] - 0s 57us/step - loss: 0.5995 -
accuracy: 0.7716
Epoch 89/100
740/740 [==============================] - 0s 49us/step - loss: 0.6313 -
accuracy: 0.7473
Epoch 90/100
740/740 [==============================] - 0s 55us/step - loss: 0.6079 -
accuracy: 0.7500
Epoch 91/100
740/740 [==============================] - 0s 47us/step - loss: 0.6151 -
accuracy: 0.7500
Epoch 92/100
740/740 [==============================] - 0s 54us/step - loss: 0.6041 -
accuracy: 0.7703
Epoch 93/100
740/740 [==============================] - 0s 50us/step - loss: 0.6120 -
```

```
accuracy: 0.7378
Epoch 94/100
740/740 [==============================] - 0s 49us/step - loss: 0.6153 -
accuracy: 0.7649
Epoch 95/100
740/740 [==============================] - 0s 50us/step - loss: 0.6090 -
accuracy: 0.7500
Epoch 96/100
740/740 [==============================] - 0s 49us/step - loss: 0.6047 -
accuracy: 0.7635
Epoch 97/100
740/740 [==============================] - 0s 47us/step - loss: 0.6034 -
accuracy: 0.7716
Epoch 98/100
740/740 [==============================] - 0s 53us/step - loss: 0.5861 -
accuracy: 0.7811
Epoch 99/100
740/740 [==============================] - 0s 49us/step - loss: 0.5971 -
accuracy: 0.7635
Epoch 100/100
740/740 [==============================] - 0s 50us/step - loss: 0.6124 -
accuracy: 0.7662
```

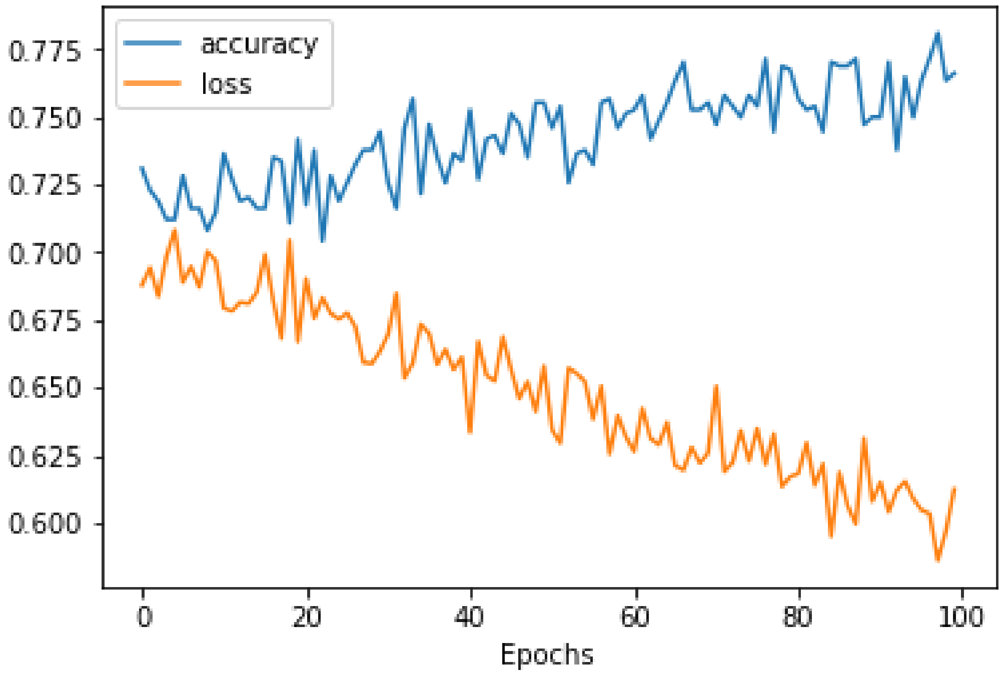
[32]:

```
val_loss_test2, val_acc_test2 = model_RNC.evaluate(X_test, Y_test)
val_acc_test.append(val_acc_test2)

print('Test accuracy:', val_acc_test2)
print('Test loss:', val_loss_test2)

val_loss_train2, val_acc_train2 = model_RNC.evaluate(X_train, Y_train)

print('Train accuracy:', val_acc_train2)
print('Train loss:', val_loss_train2)
```

```
365/365 [==============================] - 0s 41us/step
Test accuracy: 0.5479452013969421
Test loss: 1.1942667627987795
740/740 [==============================] - 0s 30us/step
Train accuracy: 0.8013513684272766
Train loss: 0.547501325124019
```

[33]:

```
pyplot.plot(history_RNC.history['accuracy'])
pyplot.plot(history_RNC.history['loss'])
pyplot.xlabel("Epochs")
pyplot.legend(['accuracy','loss'])
pyplot.show()
```



Applying Regularization (AR)

[34]:
```
model_AR = Sequential()

model_AR.add(Dense(units=25, activation='relu', input_dim=(29),␣
 ↪kernel_constraint=unit_norm(),kernel_regularizer=l2(0.01),␣
 ↪bias_regularizer=l2(0.01)))

model_AR.add(Dense(units=15, activation='relu'))
model_AR.add(Dropout(.2))

model_AR.add(Dense(units=5, activation='softmax'))

model_AR.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
model_AR.summary()
```

```
Model: "sequential_3"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 25)                750

_________________________________________________________________
dense_10 (Dense)             (None, 15)                390

_________________________________________________________________
```

```
dropout_4 (Dropout)          (None, 15)                0
_________________________________________________________________
dense_11 (Dense)             (None, 5)                 80
=================================================================
Total params: 1,220
Trainable params: 1,220
Non-trainable params: 0
_________________________________________________________________
```

[35]:
```
history_AR=model_AR.fit(X_train, Y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
740/740 [==============================] - 1s 724us/step - loss: 1.8243 -
accuracy: 0.3392
Epoch 2/100
740/740 [==============================] - 0s 57us/step - loss: 1.7348 -
accuracy: 0.4432
Epoch 3/100
740/740 [==============================] - 0s 58us/step - loss: 1.6161 -
accuracy: 0.4703
Epoch 4/100
740/740 [==============================] - 0s 55us/step - loss: 1.5195 -
accuracy: 0.4649
Epoch 5/100
740/740 [==============================] - 0s 55us/step - loss: 1.4399 -
accuracy: 0.4757
Epoch 6/100
740/740 [==============================] - 0s 59us/step - loss: 1.4107 -
accuracy: 0.5095
Epoch 7/100
740/740 [==============================] - 0s 57us/step - loss: 1.3900 -
accuracy: 0.5108
Epoch 8/100
740/740 [==============================] - 0s 58us/step - loss: 1.3848 -
accuracy: 0.4743
Epoch 9/100
740/740 [==============================] - 0s 61us/step - loss: 1.3579 -
accuracy: 0.5122
Epoch 10/100
740/740 [==============================] - 0s 57us/step - loss: 1.3582 -
accuracy: 0.5014
Epoch 11/100
740/740 [==============================] - 0s 61us/step - loss: 1.3535 -
accuracy: 0.5108
Epoch 12/100
740/740 [==============================] - 0s 62us/step - loss: 1.3452 -
accuracy: 0.5135
Epoch 13/100
```

```
740/740 [==============================] - 0s 65us/step - loss: 1.3242 -
accuracy: 0.5351
Epoch 14/100
740/740 [==============================] - 0s 65us/step - loss: 1.3313 -
accuracy: 0.5365
Epoch 15/100
740/740 [==============================] - 0s 82us/step - loss: 1.3206 -
accuracy: 0.5176
Epoch 16/100
740/740 [==============================] - 0s 62us/step - loss: 1.3082 -
accuracy: 0.5365
Epoch 17/100
740/740 [==============================] - 0s 67us/step - loss: 1.2929 -
accuracy: 0.5689
Epoch 18/100
740/740 [==============================] - 0s 61us/step - loss: 1.2960 -
accuracy: 0.5622
Epoch 19/100
740/740 [==============================] - 0s 53us/step - loss: 1.2873 -
accuracy: 0.5500
Epoch 20/100
740/740 [==============================] - 0s 59us/step - loss: 1.2810 -
accuracy: 0.5757
Epoch 21/100
740/740 [==============================] - 0s 59us/step - loss: 1.2798 -
accuracy: 0.5703
Epoch 22/100
740/740 [==============================] - 0s 59us/step - loss: 1.2745 -
accuracy: 0.5500
Epoch 23/100
740/740 [==============================] - 0s 59us/step - loss: 1.2667 -
accuracy: 0.5541
Epoch 24/100
740/740 [==============================] - 0s 61us/step - loss: 1.2568 -
accuracy: 0.5608
Epoch 25/100
740/740 [==============================] - 0s 59us/step - loss: 1.2678 -
accuracy: 0.5662
Epoch 26/100
740/740 [==============================] - 0s 81us/step - loss: 1.2565 -
accuracy: 0.5662
Epoch 27/100
740/740 [==============================] - 0s 63us/step - loss: 1.2435 -
accuracy: 0.5797
Epoch 28/100
740/740 [==============================] - 0s 67us/step - loss: 1.2500 -
accuracy: 0.5703
Epoch 29/100
```

```
740/740 [==============================] - 0s 61us/step - loss: 1.2502 -
accuracy: 0.5730
Epoch 30/100
740/740 [==============================] - 0s 58us/step - loss: 1.2420 -
accuracy: 0.5865
Epoch 31/100
740/740 [==============================] - 0s 61us/step - loss: 1.2409 -
accuracy: 0.5743
Epoch 32/100
740/740 [==============================] - 0s 61us/step - loss: 1.2430 -
accuracy: 0.5703
Epoch 33/100
740/740 [==============================] - 0s 62us/step - loss: 1.2284 -
accuracy: 0.5851
Epoch 34/100
740/740 [==============================] - 0s 63us/step - loss: 1.2292 -
accuracy: 0.5689
Epoch 35/100
740/740 [==============================] - 0s 67us/step - loss: 1.2304 -
accuracy: 0.5743
Epoch 36/100
740/740 [==============================] - 0s 65us/step - loss: 1.2316 -
accuracy: 0.5892
Epoch 37/100
740/740 [==============================] - 0s 65us/step - loss: 1.2194 -
accuracy: 0.5811
Epoch 38/100
740/740 [==============================] - 0s 66us/step - loss: 1.2237 -
accuracy: 0.5784
Epoch 39/100
740/740 [==============================] - 0s 63us/step - loss: 1.2276 -
accuracy: 0.5730
Epoch 40/100
740/740 [==============================] - 0s 62us/step - loss: 1.2266 -
accuracy: 0.5878
Epoch 41/100
740/740 [==============================] - 0s 63us/step - loss: 1.2244 -
accuracy: 0.5824
Epoch 42/100
740/740 [==============================] - 0s 63us/step - loss: 1.2250 -
accuracy: 0.5838
Epoch 43/100
740/740 [==============================] - 0s 57us/step - loss: 1.2177 -
accuracy: 0.5770
Epoch 44/100
740/740 [==============================] - 0s 67us/step - loss: 1.2081 -
accuracy: 0.5824
Epoch 45/100
```

```
740/740 [==============================] - 0s 55us/step - loss: 1.1973 -
accuracy: 0.5905
Epoch 46/100
740/740 [==============================] - 0s 62us/step - loss: 1.2147 -
accuracy: 0.5865
Epoch 47/100
740/740 [==============================] - 0s 55us/step - loss: 1.2124 -
accuracy: 0.5973
Epoch 48/100
740/740 [==============================] - 0s 57us/step - loss: 1.2078 -
accuracy: 0.5811
Epoch 49/100
740/740 [==============================] - 0s 59us/step - loss: 1.2096 -
accuracy: 0.5716
Epoch 50/100
740/740 [==============================] - 0s 58us/step - loss: 1.2151 -
accuracy: 0.5676
Epoch 51/100
740/740 [==============================] - 0s 61us/step - loss: 1.1993 -
accuracy: 0.5824
Epoch 52/100
740/740 [==============================] - 0s 67us/step - loss: 1.2165 -
accuracy: 0.5865
Epoch 53/100
740/740 [==============================] - 0s 62us/step - loss: 1.1982 -
accuracy: 0.5919
Epoch 54/100
740/740 [==============================] - 0s 63us/step - loss: 1.1982 -
accuracy: 0.5824
Epoch 55/100
740/740 [==============================] - 0s 65us/step - loss: 1.2051 -
accuracy: 0.5743
Epoch 56/100
740/740 [==============================] - 0s 53us/step - loss: 1.1988 -
accuracy: 0.5878
Epoch 57/100
740/740 [==============================] - 0s 54us/step - loss: 1.1831 -
accuracy: 0.5905
Epoch 58/100
740/740 [==============================] - 0s 53us/step - loss: 1.1997 -
accuracy: 0.5784
Epoch 59/100
740/740 [==============================] - 0s 55us/step - loss: 1.1859 -
accuracy: 0.5905
Epoch 60/100
740/740 [==============================] - 0s 53us/step - loss: 1.2030 -
accuracy: 0.5703
Epoch 61/100
```

```
740/740 [==============================] - 0s 50us/step - loss: 1.1937 -
accuracy: 0.5703
Epoch 62/100
740/740 [==============================] - 0s 55us/step - loss: 1.1918 -
accuracy: 0.5797
Epoch 63/100
740/740 [==============================] - 0s 57us/step - loss: 1.1971 -
accuracy: 0.5743
Epoch 64/100
740/740 [==============================] - 0s 58us/step - loss: 1.1935 -
accuracy: 0.5838
Epoch 65/100
740/740 [==============================] - 0s 58us/step - loss: 1.1673 -
accuracy: 0.5973
Epoch 66/100
740/740 [==============================] - 0s 53us/step - loss: 1.1912 -
accuracy: 0.5838
Epoch 67/100
740/740 [==============================] - 0s 54us/step - loss: 1.1876 -
accuracy: 0.5851
Epoch 68/100
740/740 [==============================] - 0s 59us/step - loss: 1.1927 -
accuracy: 0.5784
Epoch 69/100
740/740 [==============================] - 0s 58us/step - loss: 1.1892 -
accuracy: 0.5892
Epoch 70/100
740/740 [==============================] - 0s 59us/step - loss: 1.1718 -
accuracy: 0.5878
Epoch 71/100
740/740 [==============================] - 0s 54us/step - loss: 1.1801 -
accuracy: 0.5959
Epoch 72/100
740/740 [==============================] - 0s 53us/step - loss: 1.1887 -
accuracy: 0.5973
Epoch 73/100
740/740 [==============================] - 0s 57us/step - loss: 1.1833 -
accuracy: 0.5824
Epoch 74/100
740/740 [==============================] - 0s 58us/step - loss: 1.1813 -
accuracy: 0.5824
Epoch 75/100
740/740 [==============================] - 0s 51us/step - loss: 1.1789 -
accuracy: 0.5946
Epoch 76/100
740/740 [==============================] - 0s 58us/step - loss: 1.1711 -
accuracy: 0.6122
Epoch 77/100
```

```
740/740 [==============================] - 0s 53us/step - loss: 1.1770 -
accuracy: 0.5919
Epoch 78/100
740/740 [==============================] - 0s 53us/step - loss: 1.1730 -
accuracy: 0.6027
Epoch 79/100
740/740 [==============================] - 0s 51us/step - loss: 1.1733 -
accuracy: 0.5905
Epoch 80/100
740/740 [==============================] - 0s 54us/step - loss: 1.1798 -
accuracy: 0.5932
Epoch 81/100
740/740 [==============================] - 0s 53us/step - loss: 1.1800 -
accuracy: 0.5716
Epoch 82/100
740/740 [==============================] - 0s 57us/step - loss: 1.1722 -
accuracy: 0.5932
Epoch 83/100
740/740 [==============================] - 0s 54us/step - loss: 1.1711 -
accuracy: 0.5797
Epoch 84/100
740/740 [==============================] - 0s 54us/step - loss: 1.1628 -
accuracy: 0.5878
Epoch 85/100
740/740 [==============================] - 0s 51us/step - loss: 1.1796 -
accuracy: 0.5932
Epoch 86/100
740/740 [==============================] - 0s 59us/step - loss: 1.1687 -
accuracy: 0.6027
Epoch 87/100
740/740 [==============================] - 0s 65us/step - loss: 1.1621 -
accuracy: 0.5973
Epoch 88/100
740/740 [==============================] - 0s 67us/step - loss: 1.1667 -
accuracy: 0.5878
Epoch 89/100
740/740 [==============================] - 0s 78us/step - loss: 1.1571 -
accuracy: 0.5986
Epoch 90/100
740/740 [==============================] - 0s 62us/step - loss: 1.1745 -
accuracy: 0.5892
Epoch 91/100
740/740 [==============================] - 0s 61us/step - loss: 1.1662 -
accuracy: 0.5878
Epoch 92/100
740/740 [==============================] - 0s 62us/step - loss: 1.1596 -
accuracy: 0.5919
Epoch 93/100
```

```
740/740 [==============================] - 0s 54us/step - loss: 1.1641 -
accuracy: 0.6027
Epoch 94/100
740/740 [==============================] - 0s 61us/step - loss: 1.1546 -
accuracy: 0.5824
Epoch 95/100
740/740 [==============================] - 0s 65us/step - loss: 1.1751 -
accuracy: 0.5892
Epoch 96/100
740/740 [==============================] - 0s 63us/step - loss: 1.1544 -
accuracy: 0.5932
Epoch 97/100
740/740 [==============================] - 0s 63us/step - loss: 1.1538 -
accuracy: 0.6054
Epoch 98/100
740/740 [==============================] - 0s 59us/step - loss: 1.1559 -
accuracy: 0.5878
Epoch 99/100
740/740 [==============================] - 0s 58us/step - loss: 1.1580 -
accuracy: 0.5946
Epoch 100/100
740/740 [==============================] - 0s 54us/step - loss: 1.1549 -
accuracy: 0.5932
```

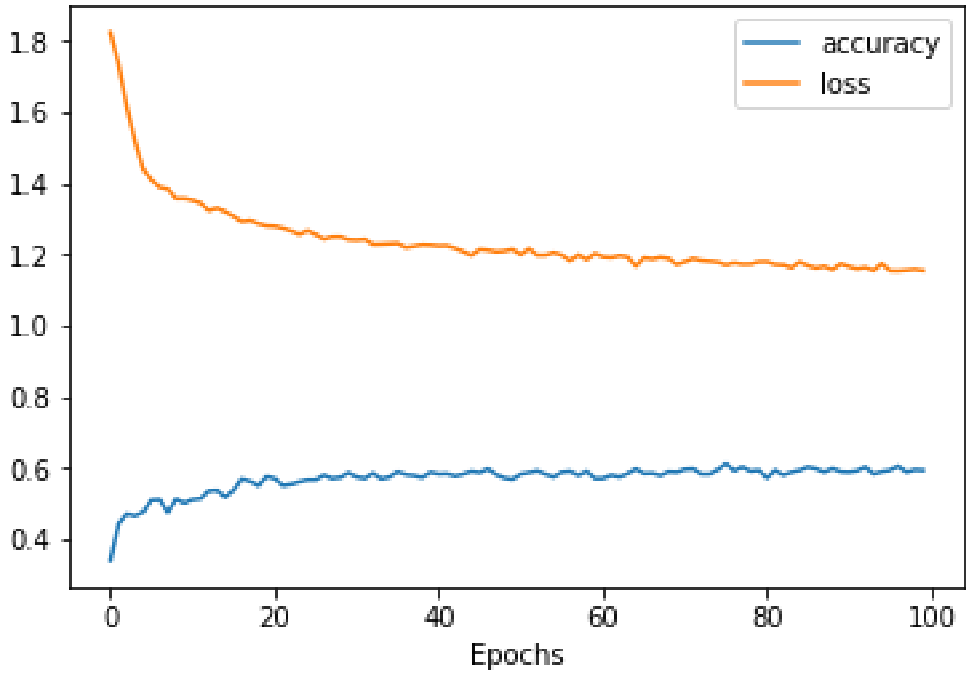
[36]:

```
val_loss_test3, val_acc_test3 = model_AR.evaluate(X_test, Y_test)
val_acc_test.append(val_acc_test3)
print('Test accuracy:', val_acc_test3)
print('Test loss:', val_loss_test3)

val_loss_train3, val_acc_train3 = model_AR.evaluate(X_train, Y_train)

print('Train accuracy:', val_acc_train3)
print('Train loss:', val_loss_train3)
```

```
365/365 [==============================] - 0s 284us/step
Test accuracy: 0.5123287439346313
Test loss: 1.2571685748557522
740/740 [==============================] - 0s 38us/step
Train accuracy: 0.587837815284729
Train loss: 1.131091752889994
```

[37]:

```
pyplot.plot(history_AR.history['accuracy'])
pyplot.plot(history_AR.history['loss'])
pyplot.xlabel("Epochs")
pyplot.legend(['accuracy','loss'])
pyplot.show()
```



Adding Dropout Layers (ADL)

[38]:
```
model_ADL = Sequential()

model_ADL.add(Dense(units=25, activation='relu', input_dim=(29),␣
 ↪kernel_constraint=unit_norm(),kernel_regularizer=l2(0.01),␣
 ↪bias_regularizer=l2(0.01)))
model_ADL.add(Dropout(.5))

model_ADL.add(Dense(units=15, activation='relu'))
model_ADL.add(Dropout(.4))

model_ADL.add(Dense(units=5, activation='softmax'))

model_ADL.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
model_ADL.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 25)                750
_________________________________________________________________
dropout_5 (Dropout)          (None, 25)                0
```

```
-----------------------------------------------------------------
dense_13 (Dense)             (None, 15)                390
-----------------------------------------------------------------
dropout_6 (Dropout)          (None, 15)                0
-----------------------------------------------------------------
dense_14 (Dense)             (None, 5)                 80
=================================================================
Total params: 1,220
Trainable params: 1,220
Non-trainable params: 0
-----------------------------------------------------------------
```

[39]:
```
history_ADL=model_ADL.fit(X_train, Y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
740/740 [==============================] - 1s 695us/step - loss: 1.9078 -
accuracy: 0.1270
Epoch 2/100
740/740 [==============================] - 0s 61us/step - loss: 1.8416 -
accuracy: 0.2770
Epoch 3/100
740/740 [==============================] - 0s 61us/step - loss: 1.8002 -
accuracy: 0.3676
Epoch 4/100
740/740 [==============================] - 0s 62us/step - loss: 1.7456 -
accuracy: 0.4257
Epoch 5/100
740/740 [==============================] - 0s 59us/step - loss: 1.7022 -
accuracy: 0.4486
Epoch 6/100
740/740 [==============================] - 0s 62us/step - loss: 1.6436 -
accuracy: 0.4514
Epoch 7/100
740/740 [==============================] - 0s 66us/step - loss: 1.5734 -
accuracy: 0.4824
Epoch 8/100
740/740 [==============================] - 0s 66us/step - loss: 1.5483 -
accuracy: 0.4554
Epoch 9/100
740/740 [==============================] - 0s 61us/step - loss: 1.5114 -
accuracy: 0.4392
Epoch 10/100
740/740 [==============================] - 0s 86us/step - loss: 1.4522 -
accuracy: 0.4973
Epoch 11/100
740/740 [==============================] - 0s 63us/step - loss: 1.4745 -
accuracy: 0.4514
Epoch 12/100
```

```
740/740 [==============================] - 0s 65us/step - loss: 1.4712 -
accuracy: 0.4649
Epoch 13/100
740/740 [==============================] - 0s 61us/step - loss: 1.4474 -
accuracy: 0.4514
Epoch 14/100
740/740 [==============================] - 0s 62us/step - loss: 1.4350 -
accuracy: 0.4622
Epoch 15/100
740/740 [==============================] - 0s 62us/step - loss: 1.4105 -
accuracy: 0.4797
Epoch 16/100
740/740 [==============================] - 0s 63us/step - loss: 1.4071 -
accuracy: 0.4932
Epoch 17/100
740/740 [==============================] - 0s 67us/step - loss: 1.4040 -
accuracy: 0.4811
Epoch 18/100
740/740 [==============================] - 0s 66us/step - loss: 1.4251 -
accuracy: 0.4919
Epoch 19/100
740/740 [==============================] - 0s 65us/step - loss: 1.3871 -
accuracy: 0.4770
Epoch 20/100
740/740 [==============================] - 0s 63us/step - loss: 1.3836 -
accuracy: 0.5041
Epoch 21/100
740/740 [==============================] - 0s 62us/step - loss: 1.3734 -
accuracy: 0.5149
Epoch 22/100
740/740 [==============================] - 0s 67us/step - loss: 1.3811 -
accuracy: 0.4932
Epoch 23/100
740/740 [==============================] - 0s 66us/step - loss: 1.3863 -
accuracy: 0.4703
Epoch 24/100
740/740 [==============================] - 0s 65us/step - loss: 1.3590 -
accuracy: 0.4986
Epoch 25/100
740/740 [==============================] - 0s 65us/step - loss: 1.3631 -
accuracy: 0.4932
Epoch 26/100
740/740 [==============================] - 0s 70us/step - loss: 1.3581 -
accuracy: 0.5216
Epoch 27/100
740/740 [==============================] - 0s 63us/step - loss: 1.3480 -
accuracy: 0.5189
Epoch 28/100
```

```
740/740 [==============================] - 0s 65us/step - loss: 1.3593 -
accuracy: 0.5054
Epoch 29/100
740/740 [==============================] - 0s 69us/step - loss: 1.3563 -
accuracy: 0.4932
Epoch 30/100
740/740 [==============================] - 0s 67us/step - loss: 1.3566 -
accuracy: 0.5203
Epoch 31/100
740/740 [==============================] - 0s 67us/step - loss: 1.3415 -
accuracy: 0.5243
Epoch 32/100
740/740 [==============================] - 0s 63us/step - loss: 1.3477 -
accuracy: 0.4919
Epoch 33/100
740/740 [==============================] - 0s 69us/step - loss: 1.3339 -
accuracy: 0.5284
Epoch 34/100
740/740 [==============================] - 0s 70us/step - loss: 1.3333 -
accuracy: 0.5243
Epoch 35/100
740/740 [==============================] - 0s 70us/step - loss: 1.3598 -
accuracy: 0.5081
Epoch 36/100
740/740 [==============================] - 0s 65us/step - loss: 1.3273 -
accuracy: 0.5257
Epoch 37/100
740/740 [==============================] - 0s 65us/step - loss: 1.3305 -
accuracy: 0.5203
Epoch 38/100
740/740 [==============================] - 0s 67us/step - loss: 1.3148 -
accuracy: 0.5446
Epoch 39/100
740/740 [==============================] - 0s 66us/step - loss: 1.3308 -
accuracy: 0.5338
Epoch 40/100
740/740 [==============================] - 0s 62us/step - loss: 1.3383 -
accuracy: 0.5270
Epoch 41/100
740/740 [==============================] - 0s 63us/step - loss: 1.3099 -
accuracy: 0.5446
Epoch 42/100
740/740 [==============================] - 0s 66us/step - loss: 1.3372 -
accuracy: 0.5351
Epoch 43/100
740/740 [==============================] - 0s 67us/step - loss: 1.3248 -
accuracy: 0.5122
Epoch 44/100
```

```
740/740 [==============================] - 0s 58us/step - loss: 1.3210 -
accuracy: 0.5311
Epoch 45/100
740/740 [==============================] - 0s 70us/step - loss: 1.3010 -
accuracy: 0.5392
Epoch 46/100
740/740 [==============================] - 0s 59us/step - loss: 1.2942 -
accuracy: 0.5622
Epoch 47/100
740/740 [==============================] - 0s 65us/step - loss: 1.3000 -
accuracy: 0.5608
Epoch 48/100
740/740 [==============================] - 0s 70us/step - loss: 1.3042 -
accuracy: 0.5311
Epoch 49/100
740/740 [==============================] - 0s 65us/step - loss: 1.3032 -
accuracy: 0.5419
Epoch 50/100
740/740 [==============================] - 0s 61us/step - loss: 1.3145 -
accuracy: 0.5649
Epoch 51/100
740/740 [==============================] - 0s 62us/step - loss: 1.3052 -
accuracy: 0.5459
Epoch 52/100
740/740 [==============================] - 0s 63us/step - loss: 1.3186 -
accuracy: 0.5459
Epoch 53/100
740/740 [==============================] - 0s 65us/step - loss: 1.3168 -
accuracy: 0.5338
Epoch 54/100
740/740 [==============================] - 0s 67us/step - loss: 1.3102 -
accuracy: 0.5297
Epoch 55/100
740/740 [==============================] - 0s 59us/step - loss: 1.2902 -
accuracy: 0.5405
Epoch 56/100
740/740 [==============================] - 0s 62us/step - loss: 1.3084 -
accuracy: 0.5419
Epoch 57/100
740/740 [==============================] - 0s 63us/step - loss: 1.3081 -
accuracy: 0.5176
Epoch 58/100
740/740 [==============================] - 0s 66us/step - loss: 1.2823 -
accuracy: 0.5554
Epoch 59/100
740/740 [==============================] - 0s 61us/step - loss: 1.2801 -
accuracy: 0.5432
Epoch 60/100
```

```
740/740 [==============================] - 0s 66us/step - loss: 1.2944 -
accuracy: 0.5486
Epoch 61/100
740/740 [==============================] - 0s 57us/step - loss: 1.2950 -
accuracy: 0.5365
Epoch 62/100
740/740 [==============================] - 0s 55us/step - loss: 1.3026 -
accuracy: 0.5392
Epoch 63/100
740/740 [==============================] - 0s 66us/step - loss: 1.3069 -
accuracy: 0.5527
Epoch 64/100
740/740 [==============================] - 0s 63us/step - loss: 1.3049 -
accuracy: 0.5622
Epoch 65/100
740/740 [==============================] - 0s 65us/step - loss: 1.2891 -
accuracy: 0.5432
Epoch 66/100
740/740 [==============================] - 0s 62us/step - loss: 1.2927 -
accuracy: 0.5689
Epoch 67/100
740/740 [==============================] - 0s 67us/step - loss: 1.2993 -
accuracy: 0.5581
Epoch 68/100
740/740 [==============================] - 0s 63us/step - loss: 1.3084 -
accuracy: 0.5284
Epoch 69/100
740/740 [==============================] - 0s 62us/step - loss: 1.3039 -
accuracy: 0.5459
Epoch 70/100
740/740 [==============================] - 0s 65us/step - loss: 1.2887 -
accuracy: 0.5486
Epoch 71/100
740/740 [==============================] - 0s 63us/step - loss: 1.2908 -
accuracy: 0.5662
Epoch 72/100
740/740 [==============================] - 0s 66us/step - loss: 1.2932 -
accuracy: 0.5662
Epoch 73/100
740/740 [==============================] - 0s 77us/step - loss: 1.2981 -
accuracy: 0.5446
Epoch 74/100
740/740 [==============================] - 0s 109us/step - loss: 1.2867 -
accuracy: 0.5446
Epoch 75/100
740/740 [==============================] - 0s 85us/step - loss: 1.2710 -
accuracy: 0.5568
Epoch 76/100
```

```
740/740 [==============================] - 0s 86us/step - loss: 1.2878 -
accuracy: 0.5486
Epoch 77/100
740/740 [==============================] - 0s 100us/step - loss: 1.2851 -
accuracy: 0.5608
Epoch 78/100
740/740 [==============================] - 0s 85us/step - loss: 1.2888 -
accuracy: 0.5459
Epoch 79/100
740/740 [==============================] - 0s 82us/step - loss: 1.2769 -
accuracy: 0.5595
Epoch 80/100
740/740 [==============================] - 0s 112us/step - loss: 1.2962 -
accuracy: 0.5622
Epoch 81/100
740/740 [==============================] - 0s 66us/step - loss: 1.2994 -
accuracy: 0.5446
Epoch 82/100
740/740 [==============================] - 0s 71us/step - loss: 1.2861 -
accuracy: 0.5649
Epoch 83/100
740/740 [==============================] - 0s 65us/step - loss: 1.2698 -
accuracy: 0.5541
Epoch 84/100
740/740 [==============================] - 0s 70us/step - loss: 1.3029 -
accuracy: 0.5541
Epoch 85/100
740/740 [==============================] - 0s 73us/step - loss: 1.2704 -
accuracy: 0.5527
Epoch 86/100
740/740 [==============================] - 0s 65us/step - loss: 1.2849 -
accuracy: 0.5581
Epoch 87/100
740/740 [==============================] - 0s 63us/step - loss: 1.2757 -
accuracy: 0.5622
Epoch 88/100
740/740 [==============================] - 0s 67us/step - loss: 1.2816 -
accuracy: 0.5459
Epoch 89/100
740/740 [==============================] - 0s 70us/step - loss: 1.2823 -
accuracy: 0.5486
Epoch 90/100
740/740 [==============================] - 0s 61us/step - loss: 1.2842 -
accuracy: 0.5527
Epoch 91/100
740/740 [==============================] - 0s 55us/step - loss: 1.2919 -
accuracy: 0.5365
Epoch 92/100
```

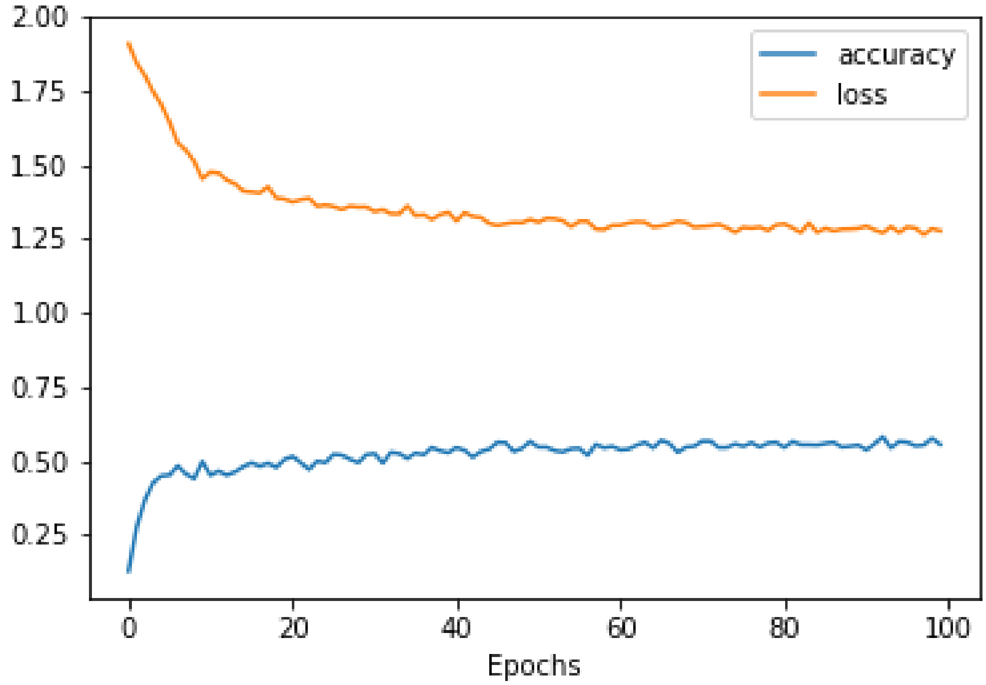
```
740/740 [==============================] - 0s 55us/step - loss: 1.2794 -
accuracy: 0.5568
Epoch 93/100
740/740 [==============================] - 0s 54us/step - loss: 1.2688 -
accuracy: 0.5811
Epoch 94/100
740/740 [==============================] - 0s 58us/step - loss: 1.2904 -
accuracy: 0.5432
Epoch 95/100
740/740 [==============================] - 0s 61us/step - loss: 1.2704 -
accuracy: 0.5649
Epoch 96/100
740/740 [==============================] - 0s 66us/step - loss: 1.2894 -
accuracy: 0.5622
Epoch 97/100
740/740 [==============================] - 0s 66us/step - loss: 1.2849 -
accuracy: 0.5500
Epoch 98/100
740/740 [==============================] - 0s 69us/step - loss: 1.2635 -
accuracy: 0.5527
Epoch 99/100
740/740 [==============================] - 0s 63us/step - loss: 1.2834 -
accuracy: 0.5757
Epoch 100/100
740/740 [==============================] - 0s 59us/step - loss: 1.2763 -
accuracy: 0.5554
```

[40]:
```
val_loss_test4, val_acc_test4 = model_ADL.evaluate(X_test, Y_test)
val_acc_test.append(val_acc_test4)
print('Test accuracy:', val_acc_test4)
print('Test loss:', val_loss_test4)

val_loss_train4, val_acc_train4 = model_ADL.evaluate(X_train, Y_train)

print('Train accuracy:', val_acc_train4)
print('Train loss:', val_loss_train4)
```

```
365/365 [==============================] - 0s 325us/step
Test accuracy: 0.5287671089172363
Test loss: 1.2713906800910217
740/740 [==============================] - 0s 38us/step
Train accuracy: 0.5864864587783813
Train loss: 1.2194807629327515
```

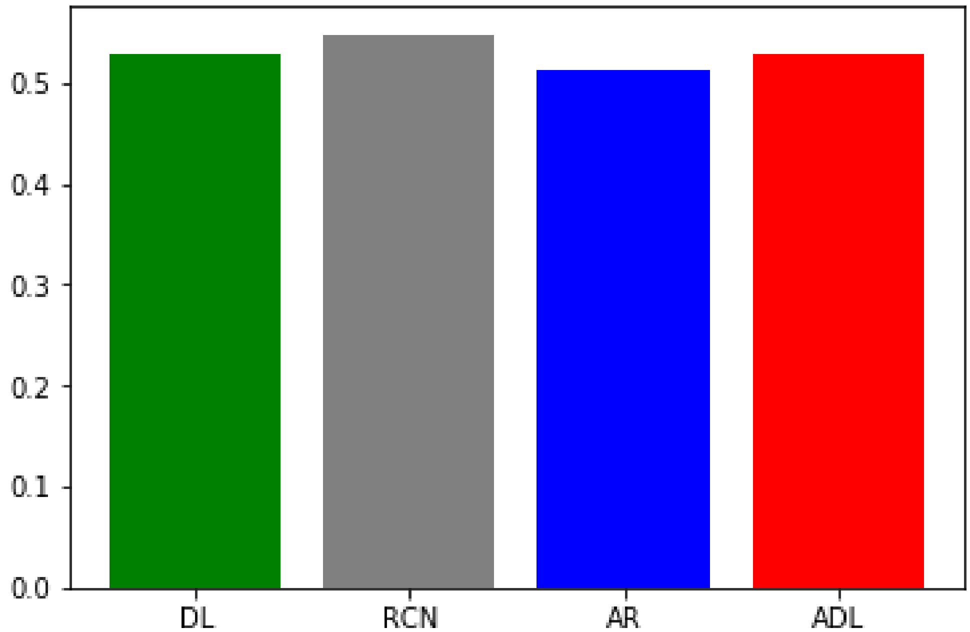
[41]:
```
pyplot.plot(history_ADL.history['accuracy'])
pyplot.plot(history_ADL.history['loss'])
pyplot.xlabel("Epochs")
```

```
pyplot.legend(['accuracy','loss'])
pyplot.show()
```



[43]:
```
colors1 = ['green', 'grey', 'blue', 'red']
names1 =  ['DL', 'RCN', 'AR', 'ADL']
pyplot.bar(names1, val_acc_test,color=colors1)
```

[43]: <BarContainer object of 4 artists>



Combining different representation approaches

```
[44]: # Use Input layers, specify input shape
def model_emb_def (no_of_unique_cat,inp_num_data):

    #Jeremy Howard provides the following rule of thumb; embedding size =␣
 ↪min(50, number of categories/2).
    embedding_size = min(np.ceil((no_of_unique_cat)/2), 50 )
    embedding_size = int(embedding_size)

    inp_cat_data = keras.layers.Input(shape=(no_of_unique_cat,))
    # Bind nulti_hot to embedding layer
    emb = keras.layers.Embedding(input_dim=no_of_unique_cat,␣
 ↪output_dim=embedding_size)(inp_cat_data)

    # Also we need flatten embedded output
    # otherwise it's not possible to concatenate it with inp_num_data
    flatten = keras.layers.Flatten()(emb)
    # Concatenate two layers
    conc = keras.layers.Concatenate()([flatten, inp_num_data])
    dense1 = keras.layers.Dense(50, activation=tf.nn.relu, )(conc)

    x= keras.layers.Dense(units=25, activation='relu')(dense1)
    x= keras.layers.Dropout(.2)(x)
    x= keras.layers.Dense(units=15, activation='relu')(x)
```

```
    x= keras.layers.Dropout(.2)(x)
    x= keras.layers.Dense(units=10, activation='relu')(x)

    # Creating output layer
    #out = keras.layers.Dense(5, activation=tf.keras.activations.
↪softmax)(dense1)
    out = keras.layers.Dense(5, activation=tf.keras.activations.softmax)(x)
    model_def = keras.Model(inputs=[inp_cat_data, inp_num_data], outputs=out)

    return model_def
```

Combining *tf_idf* and *specific quoted article* feature representations (TF-IDF+SA)

[45]:
```
# Use Input layers, specify input shape
no_of_unique_cat= categorical_article_df.shape[1]-1
inp_num_data = keras.layers.Input(shape=(X.shape[1],))

# Creating the model
model_Emb = model_emb_def(no_of_unique_cat,inp_num_data)
model_Emb.summary()
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_2 (InputLayer)            [(None, 244)]        0

________________________________________________________________________________
__________________
embedding (Embedding)           (None, 244, 50)      12200       input_2[0][0]

________________________________________________________________________________
__________________
flatten (Flatten)               (None, 12200)        0           embedding[0][0]

________________________________________________________________________________
__________________
input_1 (InputLayer)            [(None, 29)]         0

________________________________________________________________________________
__________________
concatenate (Concatenate)       (None, 12229)        0           flatten[0][0]
                                                                 input_1[0][0]

________________________________________________________________________________
__________________
dense (Dense)                   (None, 50)           611500
concatenate[0][0]

________________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 25)           1275        dense[0][0]
```

```
--------------------------------------------------------------------------------
------------------
dropout (Dropout)               (None, 25)          0           dense_1[0][0]
--------------------------------------------------------------------------------
------------------
dense_2 (Dense)                 (None, 15)          390         dropout[0][0]
--------------------------------------------------------------------------------
------------------
dropout_1 (Dropout)             (None, 15)          0           dense_2[0][0]
--------------------------------------------------------------------------------
------------------
dense_3 (Dense)                 (None, 10)          160         dropout_1[0][0]
--------------------------------------------------------------------------------
------------------
dense_4 (Dense)                 (None, 5)           55          dense_3[0][0]
================================================================================
==================
Total params: 625,580
Trainable params: 625,580
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
```

[46]:
```
model_Emb.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
```

[47]:
```
df_Cat_GDPR_Article_Df= pd.read_csv('Data/Categorical_GDPR_Article_Df.csv')
df_id_names= df_Tf_Idf.iloc[:,-2:]
df_Cat_GDPR_Art_Df_new1=pd.concat([df_Cat_GDPR_Article_Df, df_Tf_Idf], axis=1).
 ↪reindex(df_Cat_GDPR_Article_Df.index)
df_Cat_GDPR_Art_Df_new1['Fine_binned1'] = pd.
 ↪cut(df_Cat_GDPR_Art_Df_new1['Fine'], bins=cut_bins, labels=False)
df_Cat_GDPR_Art_Df_new1['Fine_binned1'] =␣
 ↪df_Cat_GDPR_Art_Df_new1['Fine_binned1'].astype(np.int64)
```

[48]:
```
# tail of the dataset
df_Cat_GDPR_Art_Df_new1.tail(5)
```

[48]:
```
      13 GDPR  5 GDPR  14 GDPR  5 (1) a) GDPR  6 GDPR  5 (1) c) GDPR  \
1100        0       0        0              0       0              0
1101        0       0        0              0       0              0
1102        0       0        0              0       0              0
1103        0       0        0              0       0              0
1104        0       0        0              0       0              0

      6 (1) GDPR  5 (1) b) GDPR  15 GDPR  32 GDPR     …     receiv  \
1100           0              0        0        0     …        0.0
```

```
1101          0            0        0        0    ...         0.0
1102          0            0        0        0    ...         0.0
1103          0            0        0        0    ...         0.0
1104          0            0        0        0    ...         0.0

       request  secur   spanish  surveil  system    violat   Fine         Id  \
1100  0.252572    0.0  0.233186      0.0     0.0  0.000000   4200  ETid-1143
1101  0.000000    0.0  0.297958      0.0     0.0  0.000000  16000  ETid-1144
1102  0.476385    0.0  0.000000      0.0     0.0  0.424414   4000  ETid-1145
1103  0.433148    0.0  0.000000      0.0     0.0  0.000000  10000  ETid-1146
1104  0.000000    0.0  0.339194      0.0     0.0  0.000000   1500  ETid-1147

      Fine_binned1
1100             0
1101             1
1102             0
1103             1
1104             0

[5 rows x 277 columns]
```

[49]:
```
X1 = df_Cat_GDPR_Art_Df_new1.iloc[:,0:len(df_Cat_GDPR_Art_Df_new1.columns)-3]
Y1 = df_Cat_GDPR_Art_Df_new1['Fine_binned1']
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size=0.33,␣
 ↪random_state=42)
```

[50]:
```
X1_train_Cat= X1_train.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]
X1_test_Cat= X1_test.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]

X1_train_Num= X1_train.iloc[:,-len(df_Tf_Idf.columns)+2:]
X1_test_Num= X1_test.iloc[:,-len(df_Tf_Idf.columns)+2:]
```

[51]:
```
history_Emb = model_Emb.fit([X1_train_Cat, X1_train_Num], Y1_train,epochs=100,␣
 ↪batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 926us/sample - loss: 1.3205 -
accuracy: 0.3892
Epoch 2/100
740/740 [==============================] - 0s 276us/sample - loss: 1.2547 -
accuracy: 0.3824
Epoch 3/100
740/740 [==============================] - 0s 272us/sample - loss: 1.1993 -
```

```
accuracy: 0.4135
Epoch 4/100
740/740 [==============================] - 0s 279us/sample - loss: 1.1859 -
accuracy: 0.4365
Epoch 5/100
740/740 [==============================] - 0s 275us/sample - loss: 1.1700 -
accuracy: 0.4581
Epoch 6/100
740/740 [==============================] - 0s 299us/sample - loss: 1.1587 -
accuracy: 0.4743
Epoch 7/100
740/740 [==============================] - 0s 305us/sample - loss: 1.1421 -
accuracy: 0.4824
Epoch 8/100
740/740 [==============================] - 0s 276us/sample - loss: 1.1369 -
accuracy: 0.5135
Epoch 9/100
740/740 [==============================] - 0s 276us/sample - loss: 1.0917 -
accuracy: 0.4919
Epoch 10/100
740/740 [==============================] - 0s 271us/sample - loss: 1.0743 -
accuracy: 0.5176
Epoch 11/100
740/740 [==============================] - 0s 271us/sample - loss: 1.0498 -
accuracy: 0.5473
Epoch 12/100
740/740 [==============================] - 0s 272us/sample - loss: 1.0011 -
accuracy: 0.5635
Epoch 13/100
740/740 [==============================] - 0s 271us/sample - loss: 0.9912 -
accuracy: 0.5581
Epoch 14/100
740/740 [==============================] - 0s 278us/sample - loss: 0.9665 -
accuracy: 0.5662
Epoch 15/100
740/740 [==============================] - 0s 272us/sample - loss: 0.9459 -
accuracy: 0.5892
Epoch 16/100
740/740 [==============================] - 0s 271us/sample - loss: 0.9262 -
accuracy: 0.5973
Epoch 17/100
740/740 [==============================] - 0s 272us/sample - loss: 0.9265 -
accuracy: 0.6162
Epoch 18/100
740/740 [==============================] - 0s 282us/sample - loss: 0.9234 -
accuracy: 0.5932
Epoch 19/100
740/740 [==============================] - 0s 283us/sample - loss: 0.8824 -
```

```
accuracy: 0.6189
Epoch 20/100
740/740 [==============================] - 0s 310us/sample - loss: 0.8516 -
accuracy: 0.6270
Epoch 21/100
740/740 [==============================] - 0s 305us/sample - loss: 0.8811 -
accuracy: 0.6338
Epoch 22/100
740/740 [==============================] - 0s 271us/sample - loss: 0.8474 -
accuracy: 0.6392
Epoch 23/100
740/740 [==============================] - 0s 306us/sample - loss: 0.8473 -
accuracy: 0.6419
Epoch 24/100
740/740 [==============================] - 0s 272us/sample - loss: 0.8270 -
accuracy: 0.6662
Epoch 25/100
740/740 [==============================] - 0s 272us/sample - loss: 0.8118 -
accuracy: 0.6527
Epoch 26/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7849 -
accuracy: 0.6649
Epoch 27/100
740/740 [==============================] - 0s 278us/sample - loss: 0.7767 -
accuracy: 0.6689
Epoch 28/100
740/740 [==============================] - 0s 278us/sample - loss: 0.7624 -
accuracy: 0.6757
Epoch 29/100
740/740 [==============================] - 0s 271us/sample - loss: 0.7608 -
accuracy: 0.6716
Epoch 30/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7283 -
accuracy: 0.6919
Epoch 31/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7365 -
accuracy: 0.6919
Epoch 32/100
740/740 [==============================] - 0s 271us/sample - loss: 0.7469 -
accuracy: 0.6676
Epoch 33/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7194 -
accuracy: 0.6959
Epoch 34/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7142 -
accuracy: 0.6892
Epoch 35/100
740/740 [==============================] - 0s 275us/sample - loss: 0.7058 -
```

```
accuracy: 0.6946
Epoch 36/100
740/740 [==============================] - 0s 276us/sample - loss: 0.6830 -
accuracy: 0.7122
Epoch 37/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6872 -
accuracy: 0.6986
Epoch 38/100
740/740 [==============================] - 0s 268us/sample - loss: 0.6616 -
accuracy: 0.7054
Epoch 39/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6590 -
accuracy: 0.7081
Epoch 40/100
740/740 [==============================] - 0s 272us/sample - loss: 0.6479 -
accuracy: 0.7095
Epoch 41/100
740/740 [==============================] - 0s 272us/sample - loss: 0.6350 -
accuracy: 0.7162
Epoch 42/100
740/740 [==============================] - 0s 270us/sample - loss: 0.6400 -
accuracy: 0.7203
Epoch 43/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6030 -
accuracy: 0.7446
Epoch 44/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6211 -
accuracy: 0.7297
Epoch 45/100
740/740 [==============================] - 0s 272us/sample - loss: 0.6113 -
accuracy: 0.7486
Epoch 46/100
740/740 [==============================] - 0s 279us/sample - loss: 0.6067 -
accuracy: 0.7243
Epoch 47/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5825 -
accuracy: 0.7473
Epoch 48/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5885 -
accuracy: 0.7527
Epoch 49/100
740/740 [==============================] - 0s 279us/sample - loss: 0.6132 -
accuracy: 0.7351
Epoch 50/100
740/740 [==============================] - 0s 276us/sample - loss: 0.6121 -
accuracy: 0.7473
Epoch 51/100
740/740 [==============================] - 0s 267us/sample - loss: 0.5776 -
```

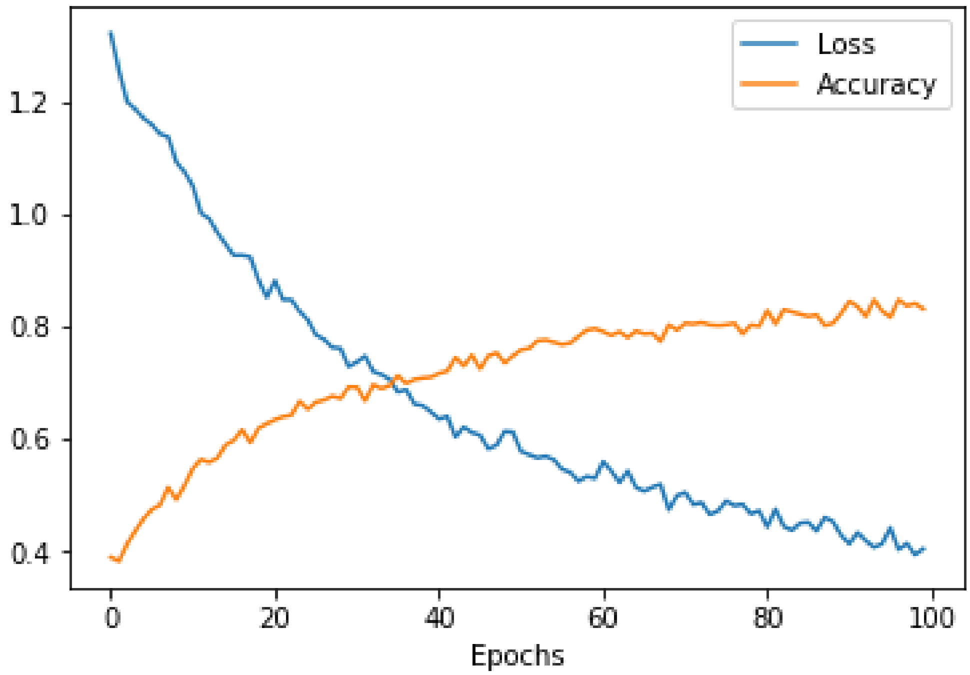
```
accuracy: 0.7581
Epoch 52/100
740/740 [==============================] - 0s 290us/sample - loss: 0.5718 -
accuracy: 0.7608
Epoch 53/100
740/740 [==============================] - 0s 299us/sample - loss: 0.5662 -
accuracy: 0.7743
Epoch 54/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5690 -
accuracy: 0.7757
Epoch 55/100
740/740 [==============================] - 0s 276us/sample - loss: 0.5630 -
accuracy: 0.7716
Epoch 56/100
740/740 [==============================] - 0s 274us/sample - loss: 0.5461 -
accuracy: 0.7676
Epoch 57/100
740/740 [==============================] - 0s 276us/sample - loss: 0.5405 -
accuracy: 0.7703
Epoch 58/100
740/740 [==============================] - 0s 283us/sample - loss: 0.5245 -
accuracy: 0.7824
Epoch 59/100
740/740 [==============================] - 0s 279us/sample - loss: 0.5332 -
accuracy: 0.7932
Epoch 60/100
740/740 [==============================] - 0s 284us/sample - loss: 0.5296 -
accuracy: 0.7959
Epoch 61/100
740/740 [==============================] - 0s 290us/sample - loss: 0.5597 -
accuracy: 0.7905
Epoch 62/100
740/740 [==============================] - 0s 321us/sample - loss: 0.5428 -
accuracy: 0.7838
Epoch 63/100
740/740 [==============================] - 0s 279us/sample - loss: 0.5221 -
accuracy: 0.7905
Epoch 64/100
740/740 [==============================] - 0s 294us/sample - loss: 0.5430 -
accuracy: 0.7797
Epoch 65/100
740/740 [==============================] - 0s 303us/sample - loss: 0.5138 -
accuracy: 0.7919
Epoch 66/100
740/740 [==============================] - 0s 275us/sample - loss: 0.5070 -
accuracy: 0.7865
Epoch 67/100
740/740 [==============================] - 0s 318us/sample - loss: 0.5139 -
```

```
accuracy: 0.7878
Epoch 68/100
740/740 [==============================] - 0s 282us/sample - loss: 0.5194 -
accuracy: 0.7730
Epoch 69/100
740/740 [==============================] - 0s 275us/sample - loss: 0.4737 -
accuracy: 0.8027
Epoch 70/100
740/740 [==============================] - 0s 283us/sample - loss: 0.4989 -
accuracy: 0.7932
Epoch 71/100
740/740 [==============================] - 0s 280us/sample - loss: 0.5046 -
accuracy: 0.8054
Epoch 72/100
740/740 [==============================] - 0s 270us/sample - loss: 0.4833 -
accuracy: 0.8041
Epoch 73/100
740/740 [==============================] - 0s 280us/sample - loss: 0.4867 -
accuracy: 0.8068
Epoch 74/100
740/740 [==============================] - 0s 267us/sample - loss: 0.4656 -
accuracy: 0.8027
Epoch 75/100
740/740 [==============================] - 0s 272us/sample - loss: 0.4717 -
accuracy: 0.8014
Epoch 76/100
740/740 [==============================] - 0s 271us/sample - loss: 0.4895 -
accuracy: 0.8027
Epoch 77/100
740/740 [==============================] - 0s 275us/sample - loss: 0.4813 -
accuracy: 0.8054
Epoch 78/100
740/740 [==============================] - 0s 268us/sample - loss: 0.4835 -
accuracy: 0.7865
Epoch 79/100
740/740 [==============================] - 0s 275us/sample - loss: 0.4665 -
accuracy: 0.8027
Epoch 80/100
740/740 [==============================] - 0s 275us/sample - loss: 0.4720 -
accuracy: 0.8000
Epoch 81/100
740/740 [==============================] - 0s 271us/sample - loss: 0.4432 -
accuracy: 0.8284
Epoch 82/100
740/740 [==============================] - 0s 268us/sample - loss: 0.4751 -
accuracy: 0.8041
Epoch 83/100
740/740 [==============================] - 0s 266us/sample - loss: 0.4444 -
```

```
accuracy: 0.8297
Epoch 84/100
740/740 [==============================] - 0s 275us/sample - loss: 0.4373 -
accuracy: 0.8257
Epoch 85/100
740/740 [==============================] - 0s 267us/sample - loss: 0.4501 -
accuracy: 0.8216
Epoch 86/100
740/740 [==============================] - 0s 263us/sample - loss: 0.4517 -
accuracy: 0.8176
Epoch 87/100
740/740 [==============================] - 0s 270us/sample - loss: 0.4359 -
accuracy: 0.8203
Epoch 88/100
740/740 [==============================] - 0s 271us/sample - loss: 0.4603 -
accuracy: 0.8014
Epoch 89/100
740/740 [==============================] - 0s 272us/sample - loss: 0.4529 -
accuracy: 0.8054
Epoch 90/100
740/740 [==============================] - 0s 270us/sample - loss: 0.4286 -
accuracy: 0.8230
Epoch 91/100
740/740 [==============================] - 0s 268us/sample - loss: 0.4131 -
accuracy: 0.8446
Epoch 92/100
740/740 [==============================] - 0s 278us/sample - loss: 0.4332 -
accuracy: 0.8351
Epoch 93/100
740/740 [==============================] - 0s 267us/sample - loss: 0.4191 -
accuracy: 0.8176
Epoch 94/100
740/740 [==============================] - 0s 266us/sample - loss: 0.4071 -
accuracy: 0.8473
Epoch 95/100
740/740 [==============================] - 0s 267us/sample - loss: 0.4132 -
accuracy: 0.8270
Epoch 96/100
740/740 [==============================] - 0s 268us/sample - loss: 0.4418 -
accuracy: 0.8162
Epoch 97/100
740/740 [==============================] - 0s 315us/sample - loss: 0.4029 -
accuracy: 0.8473
Epoch 98/100
740/740 [==============================] - 0s 266us/sample - loss: 0.4137 -
accuracy: 0.8365
Epoch 99/100
740/740 [==============================] - 0s 307us/sample - loss: 0.3937 -
```

```
accuracy: 0.8405
Epoch 100/100
740/740 [==============================] - 0s 279us/sample - loss: 0.4044 -
accuracy: 0.8311
```

[52]:
```
pyplot.plot(history_Emb.history['loss'])
pyplot.plot(history_Emb.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[53]:
```
val_loss_test5, val_acc_test5 = model_Emb.evaluate([X1_test_Cat, X1_test_Num],␣
↪Y1_test)
val_acc_test.append(val_acc_test5)

print('Test accuracy:', val_acc_test5)
print('Test loss:', val_loss_test5)

val_loss_train5, val_acc_train5 = model_Emb.evaluate([X1_train_Cat,␣
↪X1_train_Num], Y1_train)

print('Train accuracy:', val_acc_train5)
print('Train loss:', val_loss_train5)
```

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

```
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
365/365 [==============================] - 0s 445us/sample - loss: 2.0362 -
accuracy: 0.5562
Test accuracy: 0.5561644
Test loss: 2.036159560451769
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 115us/sample - loss: 0.3325 -
accuracy: 0.8838
Train accuracy: 0.88378376
Train loss: 0.33246072836824364
```

Combining *tf_idf* and *general quoted article* feature representations (TF-IDF+GA)

[54]:
```
# Use Input layers, specify input shape
no_of_unique_gen_cat= categorical_gen_article_df.shape[1]-1
inp_num_data = keras.layers.Input(shape=(X.shape[1],))

# Creating the model
model_Emb2 = model_emb_def(no_of_unique_gen_cat,inp_num_data)
model_Emb2.summary()
```

```
Model: "model_1"
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
input_4 (InputLayer)            [(None, 62)]         0
__________________________________________________________________________________
__________________
embedding_1 (Embedding)         (None, 62, 31)       1922        input_4[0][0]
__________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 1922)         0
embedding_1[0][0]
__________________________________________________________________________________
__________________
input_3 (InputLayer)            [(None, 29)]         0
__________________________________________________________________________________
__________________
concatenate_1 (Concatenate)     (None, 1951)         0           flatten_1[0][0]
                                                                 input_3[0][0]
__________________________________________________________________________________
__________________
dense_5 (Dense)                 (None, 50)           97600
```

```
concatenate_1[0][0]
________________________________________________________________________________
__________________
dense_6 (Dense)                 (None, 25)           1275        dense_5[0][0]
________________________________________________________________________________
__________________
dropout_2 (Dropout)             (None, 25)           0           dense_6[0][0]
________________________________________________________________________________
__________________
dense_7 (Dense)                 (None, 15)           390         dropout_2[0][0]
________________________________________________________________________________
__________________
dropout_3 (Dropout)             (None, 15)           0           dense_7[0][0]
________________________________________________________________________________
__________________
dense_8 (Dense)                 (None, 10)           160         dropout_3[0][0]
________________________________________________________________________________
__________________
dense_9 (Dense)                 (None, 5)            55          dense_8[0][0]
================================================================================
==================
Total params: 101,402
Trainable params: 101,402
Non-trainable params: 0
________________________________________________________________________________
__________________
```

[55]:
```
model_Emb2.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
```

[56]:
```
df_Cat_Gen_GDPR_Article_Df= pd.read_csv('Data/Categorical_Gen_GDPR_Article_Df.
 ↪csv')
df_id_names= df_Tf_Idf.iloc[:,-2:]
df_Cat_Gen_GDPR_Art_Df_new1=pd.concat([df_Cat_Gen_GDPR_Article_Df, df_Tf_Idf],␣
 ↪axis=1).reindex(df_Cat_Gen_GDPR_Article_Df.index)
df_Cat_Gen_GDPR_Art_Df_new1['Fine_binned1'] = pd.
 ↪cut(df_Cat_Gen_GDPR_Art_Df_new1['Fine'], bins=cut_bins, labels=False)
df_Cat_Gen_GDPR_Art_Df_new1['Fine_binned1'] =␣
 ↪df_Cat_Gen_GDPR_Art_Df_new1['Fine_binned1'].astype(np.int64)
```

[57]:
```
# tail of the dataset
df_Cat_Gen_GDPR_Art_Df_new1.tail(5)
```

[57]:
```
      GDPR13  GDPR5  GDPR14  GDPR6  GDPR15  GDPR32  GDPR28  GDPR33  GDPR34  \
1100       0      0       0      0       0       0       0       0       0
1101       0      1       0      0       0       0       0       0       0
1102       0      0       0      0       0       0       0       0       0
```

```
1103       0      0       0      0       0       0       0       0       0
1104       0      1       0      1       0       0       0       0       0

      GDPR12      …       receiv   request  secur   spanish  surveil  \
1100       0      …          0.0  0.252572    0.0  0.233186      0.0
1101       0      …          0.0  0.000000    0.0  0.297958      0.0
1102       0      …          0.0  0.476385    0.0  0.000000      0.0
1103       0      …          0.0  0.433148    0.0  0.000000      0.0
1104       0      …          0.0  0.000000    0.0  0.339194      0.0

      system    violat   Fine         Id  Fine_binned1
1100     0.0  0.000000   4200  ETid-1143             0
1101     0.0  0.000000  16000  ETid-1144             1
1102     0.0  0.424414   4000  ETid-1145             0
1103     0.0  0.000000  10000  ETid-1146             1
1104     0.0  0.000000   1500  ETid-1147             0

[5 rows x 95 columns]
```

[58]:
```
X2 = df_Cat_Gen_GDPR_Art_Df_new1.iloc[:,0:len(df_Cat_Gen_GDPR_Art_Df_new1.
 ↪columns)-3]
Y2 = df_Cat_Gen_GDPR_Art_Df_new1['Fine_binned1'] #.iloc[:, -1]
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y2, test_size=0.33,␣
 ↪random_state=42)
```

[59]:
```
X2_train_Cat= X2_train.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]
X2_test_Cat= X2_test.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]

X2_train_Num= X2_train.iloc[:,-len(df_Tf_Idf.columns)+2:]
X2_test_Num= X2_test.iloc[:,-len(df_Tf_Idf.columns)+2:]
```

[60]:
```
history_Emb2 = model_Emb2.fit([X2_train_Cat, X2_train_Num],␣
 ↪Y2_train,epochs=100, batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 720us/sample - loss: 1.5961 -
accuracy: 0.2378
Epoch 2/100
740/740 [==============================] - 0s 137us/sample - loss: 1.5320 -
accuracy: 0.2541
Epoch 3/100
740/740 [==============================] - 0s 137us/sample - loss: 1.3963 -
accuracy: 0.3649
```

```
Epoch 4/100
740/740 [==============================] - 0s 137us/sample - loss: 1.3174 -
accuracy: 0.4095
Epoch 5/100
740/740 [==============================] - 0s 143us/sample - loss: 1.2324 -
accuracy: 0.4162
Epoch 6/100
740/740 [==============================] - 0s 142us/sample - loss: 1.1628 -
accuracy: 0.4405
Epoch 7/100
740/740 [==============================] - 0s 142us/sample - loss: 1.1498 -
accuracy: 0.4351
Epoch 8/100
740/740 [==============================] - 0s 160us/sample - loss: 1.1518 -
accuracy: 0.4000
Epoch 9/100
740/740 [==============================] - 0s 160us/sample - loss: 1.1220 -
accuracy: 0.4230
Epoch 10/100
740/740 [==============================] - 0s 144us/sample - loss: 1.1165 -
accuracy: 0.4446
Epoch 11/100
740/740 [==============================] - 0s 140us/sample - loss: 1.1017 -
accuracy: 0.4378
Epoch 12/100
740/740 [==============================] - 0s 154us/sample - loss: 1.1208 -
accuracy: 0.4797
Epoch 13/100
740/740 [==============================] - 0s 154us/sample - loss: 1.0936 -
accuracy: 0.4635
Epoch 14/100
740/740 [==============================] - 0s 150us/sample - loss: 1.0880 -
accuracy: 0.4797
Epoch 15/100
740/740 [==============================] - 0s 143us/sample - loss: 1.0940 -
accuracy: 0.4595
Epoch 16/100
740/740 [==============================] - 0s 142us/sample - loss: 1.0901 -
accuracy: 0.4730
Epoch 17/100
740/740 [==============================] - 0s 155us/sample - loss: 1.0823 -
accuracy: 0.4716
Epoch 18/100
740/740 [==============================] - 0s 160us/sample - loss: 1.0620 -
accuracy: 0.4703
Epoch 19/100
740/740 [==============================] - 0s 171us/sample - loss: 1.0719 -
accuracy: 0.4932
```

```
Epoch 20/100
740/740 [==============================] - 0s 163us/sample - loss: 1.0483 -
accuracy: 0.5149
Epoch 21/100
740/740 [==============================] - 0s 143us/sample - loss: 1.0410 -
accuracy: 0.5243
Epoch 22/100
740/740 [==============================] - 0s 139us/sample - loss: 1.0399 -
accuracy: 0.5108
Epoch 23/100
740/740 [==============================] - 0s 140us/sample - loss: 1.0304 -
accuracy: 0.5162
Epoch 24/100
740/740 [==============================] - 0s 160us/sample - loss: 1.0247 -
accuracy: 0.5176
Epoch 25/100
740/740 [==============================] - 0s 164us/sample - loss: 1.0194 -
accuracy: 0.5203
Epoch 26/100
740/740 [==============================] - 0s 150us/sample - loss: 1.0190 -
accuracy: 0.5095
Epoch 27/100
740/740 [==============================] - 0s 164us/sample - loss: 1.0374 -
accuracy: 0.5284
Epoch 28/100
740/740 [==============================] - 0s 150us/sample - loss: 0.9943 -
accuracy: 0.5324
Epoch 29/100
740/740 [==============================] - 0s 144us/sample - loss: 0.9941 -
accuracy: 0.5216
Epoch 30/100
740/740 [==============================] - 0s 136us/sample - loss: 0.9752 -
accuracy: 0.5581
Epoch 31/100
740/740 [==============================] - 0s 162us/sample - loss: 0.9878 -
accuracy: 0.5716
Epoch 32/100
740/740 [==============================] - 0s 156us/sample - loss: 0.9771 -
accuracy: 0.5595
Epoch 33/100
740/740 [==============================] - 0s 155us/sample - loss: 0.9704 -
accuracy: 0.5743
Epoch 34/100
740/740 [==============================] - 0s 144us/sample - loss: 0.9477 -
accuracy: 0.5797
Epoch 35/100
740/740 [==============================] - 0s 144us/sample - loss: 0.9461 -
accuracy: 0.5730
```

```
Epoch 36/100
740/740 [==============================] - 0s 143us/sample - loss: 0.9460 -
accuracy: 0.5838
Epoch 37/100
740/740 [==============================] - 0s 159us/sample - loss: 0.9278 -
accuracy: 0.5757
Epoch 38/100
740/740 [==============================] - 0s 156us/sample - loss: 0.9339 -
accuracy: 0.5838
Epoch 39/100
740/740 [==============================] - 0s 154us/sample - loss: 0.9180 -
accuracy: 0.5784
Epoch 40/100
740/740 [==============================] - 0s 156us/sample - loss: 0.9171 -
accuracy: 0.5973
Epoch 41/100
740/740 [==============================] - 0s 167us/sample - loss: 0.9298 -
accuracy: 0.5770
Epoch 42/100
740/740 [==============================] - 0s 158us/sample - loss: 0.9025 -
accuracy: 0.5986
Epoch 43/100
740/740 [==============================] - 0s 143us/sample - loss: 0.8892 -
accuracy: 0.6108
Epoch 44/100
740/740 [==============================] - 0s 201us/sample - loss: 0.8835 -
accuracy: 0.6135
Epoch 45/100
740/740 [==============================] - 0s 160us/sample - loss: 0.9006 -
accuracy: 0.6027
Epoch 46/100
740/740 [==============================] - 0s 148us/sample - loss: 0.9062 -
accuracy: 0.6000
Epoch 47/100
740/740 [==============================] - 0s 154us/sample - loss: 0.8648 -
accuracy: 0.6324
Epoch 48/100
740/740 [==============================] - 0s 146us/sample - loss: 0.8536 -
accuracy: 0.6270
Epoch 49/100
740/740 [==============================] - 0s 163us/sample - loss: 0.8619 -
accuracy: 0.5986
Epoch 50/100
740/740 [==============================] - 0s 160us/sample - loss: 0.8699 -
accuracy: 0.6135
Epoch 51/100
740/740 [==============================] - 0s 156us/sample - loss: 0.8603 -
accuracy: 0.6243
```

```
Epoch 52/100
740/740 [==============================] - 0s 139us/sample - loss: 0.8371 -
accuracy: 0.6432
Epoch 53/100
740/740 [==============================] - 0s 146us/sample - loss: 0.8498 -
accuracy: 0.6162
Epoch 54/100
740/740 [==============================] - 0s 139us/sample - loss: 0.8181 -
accuracy: 0.6500
Epoch 55/100
740/740 [==============================] - 0s 140us/sample - loss: 0.8288 -
accuracy: 0.6405
Epoch 56/100
740/740 [==============================] - 0s 137us/sample - loss: 0.8058 -
accuracy: 0.6703
Epoch 57/100
740/740 [==============================] - 0s 154us/sample - loss: 0.8150 -
accuracy: 0.6541
Epoch 58/100
740/740 [==============================] - 0s 146us/sample - loss: 0.8001 -
accuracy: 0.6568
Epoch 59/100
740/740 [==============================] - 0s 151us/sample - loss: 0.7955 -
accuracy: 0.6500
Epoch 60/100
740/740 [==============================] - 0s 147us/sample - loss: 0.7730 -
accuracy: 0.6797
Epoch 61/100
740/740 [==============================] - 0s 150us/sample - loss: 0.7662 -
accuracy: 0.6622
Epoch 62/100
740/740 [==============================] - 0s 160us/sample - loss: 0.7713 -
accuracy: 0.6689
Epoch 63/100
740/740 [==============================] - 0s 148us/sample - loss: 0.7782 -
accuracy: 0.6716
Epoch 64/100
740/740 [==============================] - 0s 143us/sample - loss: 0.7683 -
accuracy: 0.6743
Epoch 65/100
740/740 [==============================] - 0s 137us/sample - loss: 0.7601 -
accuracy: 0.6757
Epoch 66/100
740/740 [==============================] - 0s 163us/sample - loss: 0.7687 -
accuracy: 0.6757
Epoch 67/100
740/740 [==============================] - 0s 168us/sample - loss: 0.7638 -
accuracy: 0.6797
```

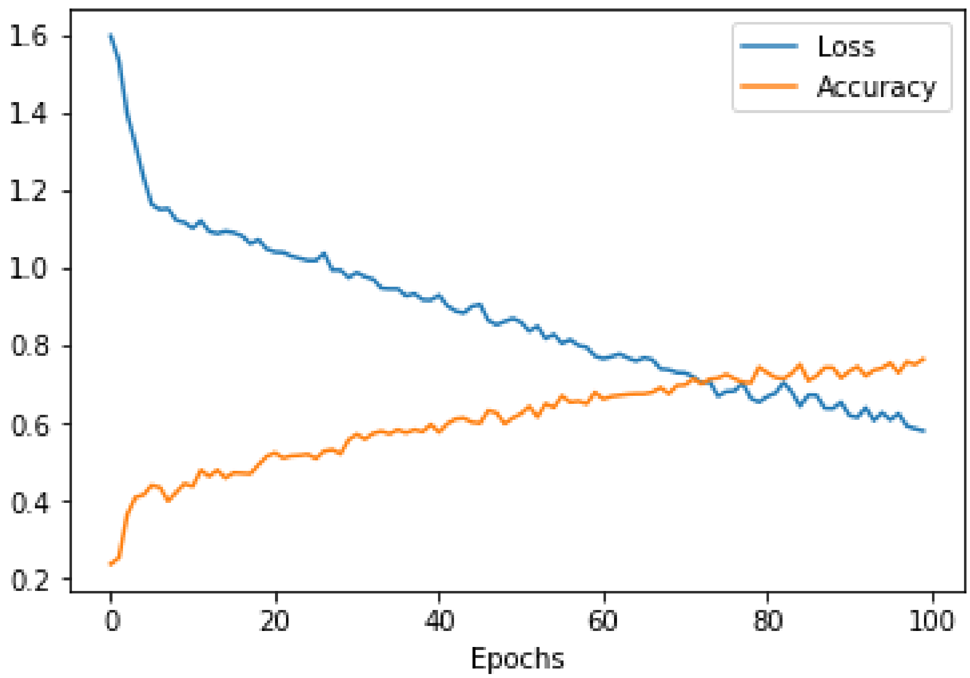
```
Epoch 68/100
740/740 [==============================] - 0s 150us/sample - loss: 0.7399 -
accuracy: 0.6919
Epoch 69/100
740/740 [==============================] - 0s 146us/sample - loss: 0.7376 -
accuracy: 0.6757
Epoch 70/100
740/740 [==============================] - 0s 147us/sample - loss: 0.7305 -
accuracy: 0.6973
Epoch 71/100
740/740 [==============================] - 0s 140us/sample - loss: 0.7294 -
accuracy: 0.6986
Epoch 72/100
740/740 [==============================] - 0s 144us/sample - loss: 0.7195 -
accuracy: 0.7149
Epoch 73/100
740/740 [==============================] - 0s 137us/sample - loss: 0.7034 -
accuracy: 0.7014
Epoch 74/100
740/740 [==============================] - 0s 147us/sample - loss: 0.7070 -
accuracy: 0.7135
Epoch 75/100
740/740 [==============================] - 0s 154us/sample - loss: 0.6698 -
accuracy: 0.7162
Epoch 76/100
740/740 [==============================] - 0s 144us/sample - loss: 0.6816 -
accuracy: 0.7257
Epoch 77/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6830 -
accuracy: 0.7149
Epoch 78/100
740/740 [==============================] - 0s 142us/sample - loss: 0.7034 -
accuracy: 0.7054
Epoch 79/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6640 -
accuracy: 0.7041
Epoch 80/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6543 -
accuracy: 0.7446
Epoch 81/100
740/740 [==============================] - 0s 139us/sample - loss: 0.6689 -
accuracy: 0.7297
Epoch 82/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6777 -
accuracy: 0.7176
Epoch 83/100
740/740 [==============================] - 0s 142us/sample - loss: 0.7051 -
accuracy: 0.7149
```

```
Epoch 84/100
740/740 [==============================] - 0s 139us/sample - loss: 0.6800 -
accuracy: 0.7284
Epoch 85/100
740/740 [==============================] - 0s 142us/sample - loss: 0.6440 -
accuracy: 0.7514
Epoch 86/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6725 -
accuracy: 0.7095
Epoch 87/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6722 -
accuracy: 0.7216
Epoch 88/100
740/740 [==============================] - 0s 142us/sample - loss: 0.6386 -
accuracy: 0.7432
Epoch 89/100
740/740 [==============================] - 0s 133us/sample - loss: 0.6378 -
accuracy: 0.7432
Epoch 90/100
740/740 [==============================] - 0s 143us/sample - loss: 0.6541 -
accuracy: 0.7162
Epoch 91/100
740/740 [==============================] - 0s 143us/sample - loss: 0.6202 -
accuracy: 0.7338
Epoch 92/100
740/740 [==============================] - 0s 142us/sample - loss: 0.6147 -
accuracy: 0.7459
Epoch 93/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6403 -
accuracy: 0.7216
Epoch 94/100
740/740 [==============================] - 0s 143us/sample - loss: 0.6070 -
accuracy: 0.7365
Epoch 95/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6273 -
accuracy: 0.7419
Epoch 96/100
740/740 [==============================] - 0s 143us/sample - loss: 0.6086 -
accuracy: 0.7554
Epoch 97/100
740/740 [==============================] - 0s 139us/sample - loss: 0.6249 -
accuracy: 0.7297
Epoch 98/100
740/740 [==============================] - 0s 140us/sample - loss: 0.5933 -
accuracy: 0.7581
Epoch 99/100
740/740 [==============================] - 0s 137us/sample - loss: 0.5863 -
accuracy: 0.7514
```

```
Epoch 100/100
740/740 [==============================] - 0s 140us/sample - loss: 0.5809 -
accuracy: 0.7649
```

[61]:

```
pyplot.plot(history_Emb2.history['loss'])
pyplot.plot(history_Emb2.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[62]:

```
val_loss_test6, val_acc_test6 = model_Emb2.evaluate([X2_test_Cat, X2_test_Num],␣
 ↪Y2_test)
val_acc_test.append(val_acc_test6)

print('Test accuracy:', val_acc_test6)
print('Test loss:', val_loss_test6)

val_loss_train6, val_acc_train6 = model_Emb2.evaluate([X2_train_Cat,␣
 ↪X2_train_Num], Y2_train)

print('Train accuracy:', val_loss_train6)
print('Train loss:', val_acc_train6)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
```

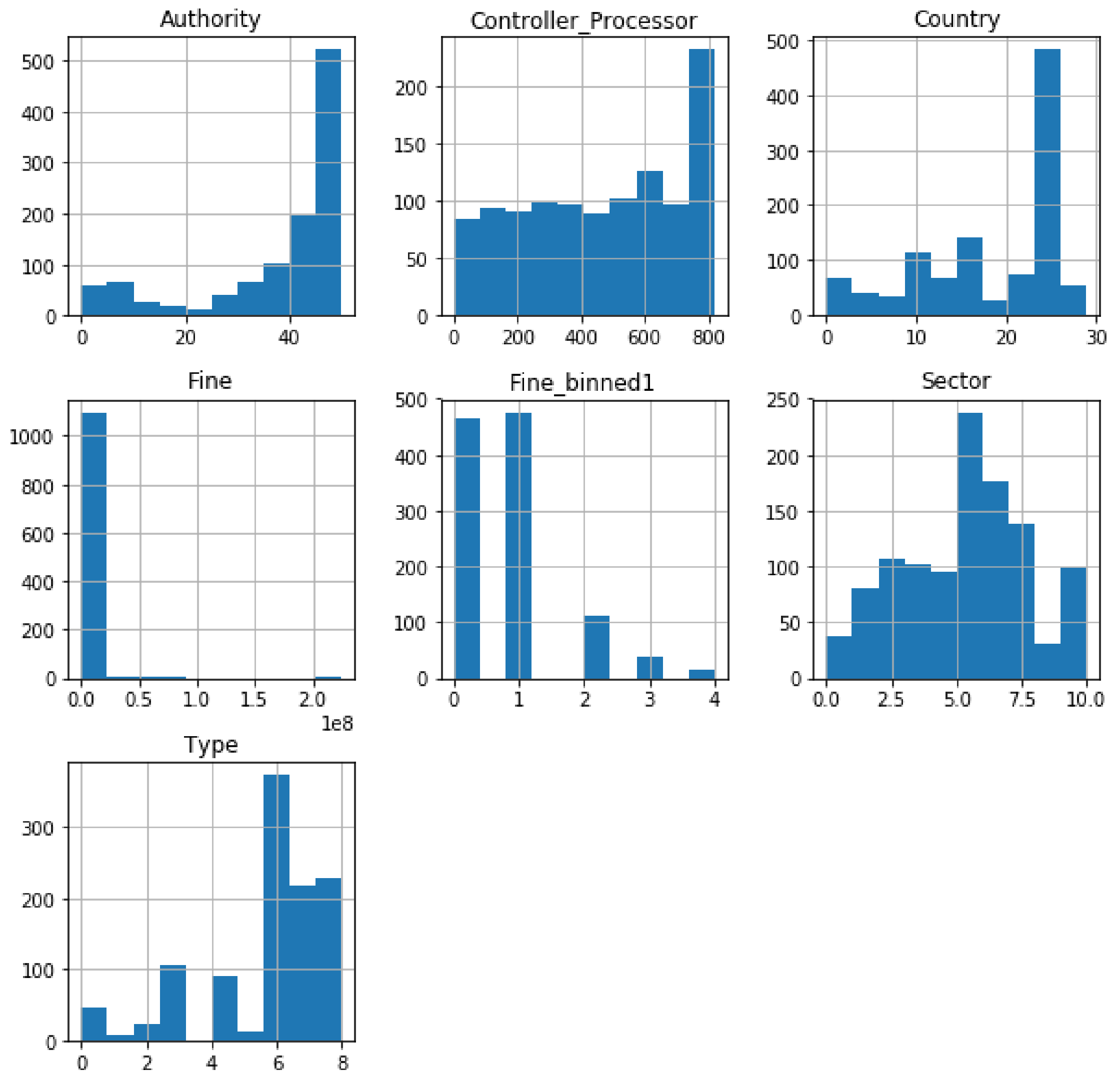
```
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
365/365 [==============================] - 0s 402us/sample - loss: 1.3363 -
accuracy: 0.5123
Test accuracy: 0.51232874
Test loss: 1.3363490493330237
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 81us/sample - loss: 0.5046 -
accuracy: 0.7973
Train accuracy: 0.5045529223777152
Train loss: 0.7972973
```

Combining *tf_idf* and *specific quoted article* feature representations with other features in the GDPR dataframe (TF-IDF+SA+OF)

[63]:
```
df_new = pd.DataFrame({"Id": df.iloc[:, 0],
                        'Country': df.iloc[:, 1],
                        'Controller_Processor': df.iloc[:, 4],
                        'Type': df.iloc[:, 6],
                        'Authority': df.iloc[:, 8],
                        'Sector': df.iloc[:, 9],
                        'Fine': df.iloc[:, 3],})

df_new.Id = df_new.Id.astype('category')
df_new.Country = df_new.Country.astype('category')
df_new.Controller_Processor = df_new.Controller_Processor.astype('category')
df_new.Type = df_new.Type.astype('category')
df_new.Authority = df_new.Authority.astype('category')
df_new.Sector = df_new.Sector.astype('category')
df_new.Fine = pd.to_numeric(df_new.Fine, downcast='integer')

df_new.Country = df_new.Country.cat.codes
df_new.Controller_Processor = df_new.Controller_Processor.cat.codes
df_new.Type = df_new.Type.cat.codes
df_new.Authority = df_new.Authority.cat.codes
df_new.Sector = df_new.Sector.cat.codes

df_new['Fine_binned1'] = pd.cut(df_new['Fine'], bins=cut_bins, labels=False)
df_new['Fine_binned1'] = df_new['Fine_binned1'].astype(np.int64)

# The correlation heatmap is produced according to the feature correlations
pyplot.figure(figsize=(7,7))
sns.heatmap(df_new.corr(),annot=True,cmap='viridis')
```

[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25a3680fa90>
```

[64]:
```
# histograms of the variables
df_new.hist(figsize=(10, 10))
pyplot.show()
```




```
[65]: from sklearn.preprocessing import StandardScaler
      df_new1 = df_new.iloc[:,1:len(df_new.columns)-2]
      sc = StandardScaler()
      df_new1 = pd.DataFrame(sc.fit_transform(df_new1))
      df_new1=pd.concat([df_new1, df_new], axis=1).reindex(df_new.index)
```

```
C:\Users\lucp9270\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\preprocessing\data.py:625: DataConversionWarning: Data with
input dtype int8, int16 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\lucp9270\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\base.py:462: DataConversionWarning: Data with input dtype int8,
int16 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

[66]:
```
df_id_names = df_Tf_Idf.iloc[:,-2:]
df_Cat_GDPR_Art_Df_new2 = pd.concat([df_Cat_GDPR_Art_Df_new1,df_new1.iloc[:,0:
 ↪5]], axis=1).reindex(df_Cat_GDPR_Art_Df_new1.index)
df_Cat_GDPR_Art_Df_new2['Fine_binned1'] = pd.
 ↪cut(df_Cat_GDPR_Art_Df_new2['Fine'], bins=cut_bins, labels=False)
df_Cat_GDPR_Art_Df_new2['Fine_binned1'] =␣
 ↪df_Cat_GDPR_Art_Df_new2['Fine_binned1'].astype(np.int64)

cols = df_Cat_GDPR_Art_Df_new2.columns.tolist()
cols = cols[:-8] + cols[-5:]+cols[len(cols)-8:len(cols)-5]

df_Cat_GDPR_Art_Df_new2 = df_Cat_GDPR_Art_Df_new2[cols]
```

[67]:
```
# tail of the dataset
df_Cat_GDPR_Art_Df_new2.tail(5)
```

[67]:
```
      13 GDPR  5 GDPR  14 GDPR  5 (1) a) GDPR  6 GDPR  5 (1) c) GDPR  \
1100        0       0        0              0       0              0
1101        0       0        0              0       0              0
1102        0       0        0              0       0              0
1103        0       0        0              0       0              0
1104        0       0        0              0       0              0

      6 (1) GDPR  5 (1) b) GDPR  15 GDPR  32 GDPR     …      violat  \
1100           0              0        0        0     …    0.000000
1101           0              0        0        0     …    0.000000
1102           0              0        0        0     …    0.424414
1103           0              0        0        0     …    0.000000
1104           0              0        0        0     …    0.000000

             0         1         2         3         4   Fine         Id  \
1100  0.877629 -1.360527 -1.369715  0.778230 -0.762759   4200  ETid-1143
1101  0.877629 -0.166206  1.085728  0.778230 -0.762759  16000  ETid-1144
1102  0.634850  0.020027 -2.842980  0.642421  0.049234   4000  ETid-1145
1103 -1.792937  0.015978 -2.842980 -2.209573 -0.762759  10000  ETid-1146
1104  0.877629 -1.388867  0.103551  0.778230 -1.574752   1500  ETid-1147

             Id  Fine_binned1
1100  ETid-1143             0
1101  ETid-1144             1
1102  ETid-1145             0
1103  ETid-1146             1
1104  ETid-1147             0

[5 rows x 284 columns]
```

[68]:
```
X3 = df_Cat_GDPR_Art_Df_new2.iloc[:,0:len(df_Cat_GDPR_Art_Df_new2.columns)-4]
Y3 = df_Cat_GDPR_Art_Df_new2['Fine_binned1'] #.iloc[:, -1]
X3_train, X3_test, Y3_train, Y3_test = train_test_split(X3, Y3, test_size=0.33,␣
↪random_state=42)

X3_train_Cat= X3_train.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]
X3_test_Cat= X3_test.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]

X3_train_Num= X3_train.iloc[:,len(df_Cat_GDPR_Article_Df.columns)+1:len(X3.
↪columns)]
X3_test_Num= X3_test.iloc[:,len(df_Cat_GDPR_Article_Df.columns)+1:len(X3.
↪columns)]
```

[69]:
```
# Use Input layers, specify input shape
no_of_unique_cat= categorical_article_df.shape[1]-1
inp_num_data = keras.layers.Input(shape=(X.shape[1]+5,))

# Creating the model
model_Emb3 = model_emb_def(no_of_unique_cat,inp_num_data)
model_Emb3.summary()
```

```
Model: "model_2"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_6 (InputLayer)            [(None, 244)]        0

________________________________________________________________________________
__________________
embedding_2 (Embedding)         (None, 244, 50)      12200       input_6[0][0]

________________________________________________________________________________
__________________
flatten_2 (Flatten)             (None, 12200)        0
embedding_2[0][0]

________________________________________________________________________________
__________________
input_5 (InputLayer)            [(None, 34)]         0

________________________________________________________________________________
__________________
concatenate_2 (Concatenate)     (None, 12234)        0           flatten_2[0][0]
                                                                 input_5[0][0]

________________________________________________________________________________
__________________
dense_10 (Dense)                (None, 50)           611750
concatenate_2[0][0]

________________________________________________________________________________
```

```
------------------
dense_11 (Dense)                (None, 25)           1275        dense_10[0][0]

--------------------------------------------------------------------------------
------------------
dropout_4 (Dropout)             (None, 25)           0           dense_11[0][0]

--------------------------------------------------------------------------------
------------------
dense_12 (Dense)                (None, 15)           390         dropout_4[0][0]

--------------------------------------------------------------------------------
------------------
dropout_5 (Dropout)             (None, 15)           0           dense_12[0][0]

--------------------------------------------------------------------------------
------------------
dense_13 (Dense)                (None, 10)           160         dropout_5[0][0]

--------------------------------------------------------------------------------
------------------
dense_14 (Dense)                (None, 5)            55          dense_13[0][0]
================================================================================
==================
Total params: 625,830
Trainable params: 625,830
Non-trainable params: 0

--------------------------------------------------------------------------------
------------------
```

[70]:
```
model_Emb3.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
history_Emb3 = model_Emb3.fit([X3_train_Cat, X3_train_Num],␣
 ↪Y3_train,epochs=100, batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 856us/sample - loss: 1.3028 -
accuracy: 0.4500
Epoch 2/100
740/740 [==============================] - 0s 276us/sample - loss: 1.1788 -
accuracy: 0.4514
Epoch 3/100
740/740 [==============================] - 0s 271us/sample - loss: 1.1744 -
accuracy: 0.4514
Epoch 4/100
740/740 [==============================] - 0s 268us/sample - loss: 1.1610 -
accuracy: 0.4514
Epoch 5/100
740/740 [==============================] - 0s 274us/sample - loss: 1.1504 -
```

```
accuracy: 0.4676
Epoch 6/100
740/740 [==============================] - 0s 271us/sample - loss: 1.1362 -
accuracy: 0.4541
Epoch 7/100
740/740 [==============================] - 0s 278us/sample - loss: 1.1397 -
accuracy: 0.4486
Epoch 8/100
740/740 [==============================] - 0s 278us/sample - loss: 1.1081 -
accuracy: 0.4581
Epoch 9/100
740/740 [==============================] - 0s 274us/sample - loss: 1.1033 -
accuracy: 0.4757
Epoch 10/100
740/740 [==============================] - 0s 272us/sample - loss: 1.0920 -
accuracy: 0.4554
Epoch 11/100
740/740 [==============================] - 0s 272us/sample - loss: 1.0722 -
accuracy: 0.4973
Epoch 12/100
740/740 [==============================] - 0s 283us/sample - loss: 1.0665 -
accuracy: 0.4905
Epoch 13/100
740/740 [==============================] - 0s 282us/sample - loss: 1.0469 -
accuracy: 0.4811
Epoch 14/100
740/740 [==============================] - 0s 302us/sample - loss: 1.0527 -
accuracy: 0.4905
Epoch 15/100
740/740 [==============================] - 0s 270us/sample - loss: 1.0090 -
accuracy: 0.5284
Epoch 16/100
740/740 [==============================] - 0s 306us/sample - loss: 0.9828 -
accuracy: 0.5419
Epoch 17/100
740/740 [==============================] - 0s 284us/sample - loss: 0.9660 -
accuracy: 0.5581
Epoch 18/100
740/740 [==============================] - 0s 305us/sample - loss: 0.9609 -
accuracy: 0.5703
Epoch 19/100
740/740 [==============================] - 0s 280us/sample - loss: 0.9481 -
accuracy: 0.5824
Epoch 20/100
740/740 [==============================] - 0s 278us/sample - loss: 0.9418 -
accuracy: 0.5743
Epoch 21/100
740/740 [==============================] - 0s 309us/sample - loss: 0.9319 -
```

```
accuracy: 0.5811
Epoch 22/100
740/740 [==============================] - 0s 291us/sample - loss: 0.9108 -
accuracy: 0.5973
Epoch 23/100
740/740 [==============================] - 0s 294us/sample - loss: 0.8682 -
accuracy: 0.5986
Epoch 24/100
740/740 [==============================] - 0s 274us/sample - loss: 0.8668 -
accuracy: 0.6095
Epoch 25/100
740/740 [==============================] - 0s 275us/sample - loss: 0.8672 -
accuracy: 0.6243
Epoch 26/100
740/740 [==============================] - 0s 272us/sample - loss: 0.8481 -
accuracy: 0.6378
Epoch 27/100
740/740 [==============================] - 0s 271us/sample - loss: 0.8261 -
accuracy: 0.6500
Epoch 28/100
740/740 [==============================] - 0s 272us/sample - loss: 0.8372 -
accuracy: 0.6311
Epoch 29/100
740/740 [==============================] - 0s 278us/sample - loss: 0.8277 -
accuracy: 0.6459
Epoch 30/100
740/740 [==============================] - 0s 272us/sample - loss: 0.7795 -
accuracy: 0.6703
Epoch 31/100
740/740 [==============================] - 0s 272us/sample - loss: 0.7718 -
accuracy: 0.6730
Epoch 32/100
740/740 [==============================] - 0s 278us/sample - loss: 0.7615 -
accuracy: 0.6905
Epoch 33/100
740/740 [==============================] - 0s 271us/sample - loss: 0.7548 -
accuracy: 0.6716
Epoch 34/100
740/740 [==============================] - 0s 274us/sample - loss: 0.7269 -
accuracy: 0.7081
Epoch 35/100
740/740 [==============================] - 0s 313us/sample - loss: 0.7299 -
accuracy: 0.6932
Epoch 36/100
740/740 [==============================] - 0s 276us/sample - loss: 0.7192 -
accuracy: 0.6973
Epoch 37/100
740/740 [==============================] - 0s 271us/sample - loss: 0.6828 -
```

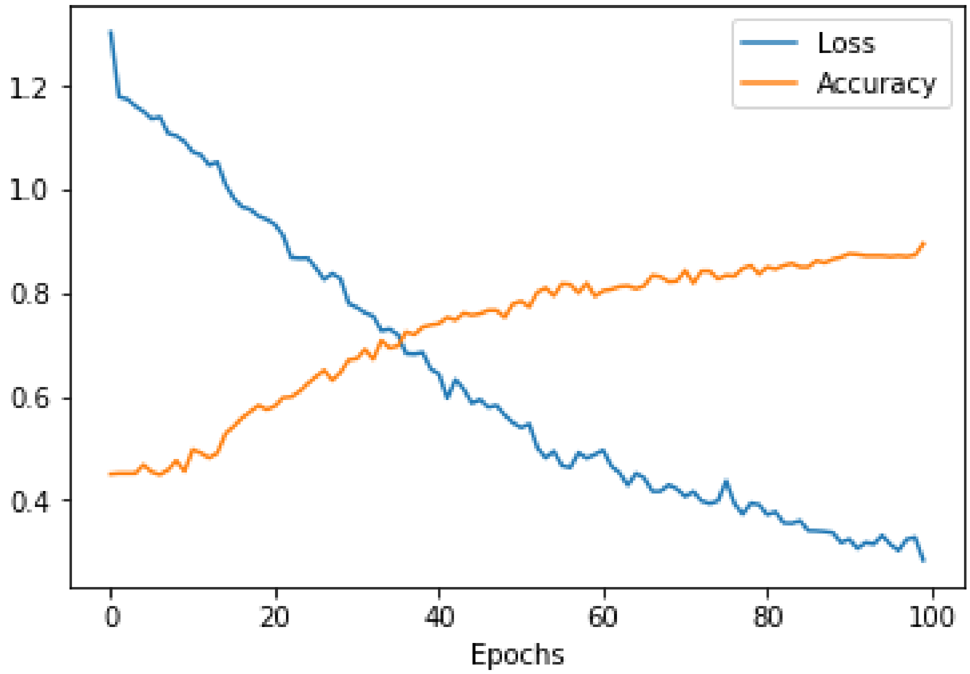
```
accuracy: 0.7243
Epoch 38/100
740/740 [==============================] - 0s 270us/sample - loss: 0.6816 -
accuracy: 0.7189
Epoch 39/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6849 -
accuracy: 0.7338
Epoch 40/100
740/740 [==============================] - 0s 276us/sample - loss: 0.6525 -
accuracy: 0.7378
Epoch 41/100
740/740 [==============================] - 0s 274us/sample - loss: 0.6428 -
accuracy: 0.7405
Epoch 42/100
740/740 [==============================] - 0s 276us/sample - loss: 0.5958 -
accuracy: 0.7527
Epoch 43/100
740/740 [==============================] - 0s 275us/sample - loss: 0.6323 -
accuracy: 0.7473
Epoch 44/100
740/740 [==============================] - 0s 274us/sample - loss: 0.6143 -
accuracy: 0.7608
Epoch 45/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5866 -
accuracy: 0.7568
Epoch 46/100
740/740 [==============================] - 0s 275us/sample - loss: 0.5937 -
accuracy: 0.7595
Epoch 47/100
740/740 [==============================] - 0s 271us/sample - loss: 0.5787 -
accuracy: 0.7662
Epoch 48/100
740/740 [==============================] - 0s 278us/sample - loss: 0.5825 -
accuracy: 0.7662
Epoch 49/100
740/740 [==============================] - 0s 271us/sample - loss: 0.5639 -
accuracy: 0.7527
Epoch 50/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5493 -
accuracy: 0.7784
Epoch 51/100
740/740 [==============================] - 0s 272us/sample - loss: 0.5395 -
accuracy: 0.7838
Epoch 52/100
740/740 [==============================] - 0s 274us/sample - loss: 0.5467 -
accuracy: 0.7730
Epoch 53/100
740/740 [==============================] - 0s 276us/sample - loss: 0.5006 -
```

```
accuracy: 0.8014
Epoch 54/100
740/740 [==============================] - 0s 271us/sample - loss: 0.4814 -
accuracy: 0.8095
Epoch 55/100
740/740 [==============================] - 0s 279us/sample - loss: 0.4939 -
accuracy: 0.7946
Epoch 56/100
740/740 [==============================] - 0s 279us/sample - loss: 0.4663 -
accuracy: 0.8176
Epoch 57/100
740/740 [==============================] - 0s 276us/sample - loss: 0.4624 -
accuracy: 0.8162
Epoch 58/100
740/740 [==============================] - 0s 288us/sample - loss: 0.4914 -
accuracy: 0.8000
Epoch 59/100
740/740 [==============================] - 0s 325us/sample - loss: 0.4804 -
accuracy: 0.8189
Epoch 60/100
740/740 [==============================] - 0s 287us/sample - loss: 0.4878 -
accuracy: 0.7932
Epoch 61/100
740/740 [==============================] - 0s 282us/sample - loss: 0.4961 -
accuracy: 0.8041
Epoch 62/100
740/740 [==============================] - 0s 280us/sample - loss: 0.4654 -
accuracy: 0.8068
Epoch 63/100
740/740 [==============================] - 0s 286us/sample - loss: 0.4523 -
accuracy: 0.8122
Epoch 64/100
740/740 [==============================] - 0s 280us/sample - loss: 0.4281 -
accuracy: 0.8135
Epoch 65/100
740/740 [==============================] - 0s 280us/sample - loss: 0.4512 -
accuracy: 0.8081
Epoch 66/100
740/740 [==============================] - 0s 278us/sample - loss: 0.4425 -
accuracy: 0.8135
Epoch 67/100
740/740 [==============================] - 0s 284us/sample - loss: 0.4166 -
accuracy: 0.8338
Epoch 68/100
740/740 [==============================] - 0s 280us/sample - loss: 0.4166 -
accuracy: 0.8311
Epoch 69/100
740/740 [==============================] - 0s 276us/sample - loss: 0.4291 -
```

```
accuracy: 0.8216
Epoch 70/100
740/740 [==============================] - 0s 271us/sample - loss: 0.4212 -
accuracy: 0.8230
Epoch 71/100
740/740 [==============================] - 0s 323us/sample - loss: 0.4065 -
accuracy: 0.8432
Epoch 72/100
740/740 [==============================] - 0s 284us/sample - loss: 0.4159 -
accuracy: 0.8189
Epoch 73/100
740/740 [==============================] - 0s 313us/sample - loss: 0.3980 -
accuracy: 0.8419
Epoch 74/100
740/740 [==============================] - 0s 267us/sample - loss: 0.3929 -
accuracy: 0.8419
Epoch 75/100
740/740 [==============================] - 0s 276us/sample - loss: 0.3979 -
accuracy: 0.8270
Epoch 76/100
740/740 [==============================] - 0s 267us/sample - loss: 0.4367 -
accuracy: 0.8338
Epoch 77/100
740/740 [==============================] - 0s 270us/sample - loss: 0.3934 -
accuracy: 0.8324
Epoch 78/100
740/740 [==============================] - 0s 267us/sample - loss: 0.3723 -
accuracy: 0.8473
Epoch 79/100
740/740 [==============================] - 0s 266us/sample - loss: 0.3935 -
accuracy: 0.8527
Epoch 80/100
740/740 [==============================] - 0s 274us/sample - loss: 0.3906 -
accuracy: 0.8365
Epoch 81/100
740/740 [==============================] - 0s 267us/sample - loss: 0.3713 -
accuracy: 0.8500
Epoch 82/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3768 -
accuracy: 0.8459
Epoch 83/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3554 -
accuracy: 0.8527
Epoch 84/100
740/740 [==============================] - 0s 266us/sample - loss: 0.3546 -
accuracy: 0.8568
Epoch 85/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3595 -
```

```
accuracy: 0.8500
Epoch 86/100
740/740 [==============================] - 0s 270us/sample - loss: 0.3404 -
accuracy: 0.8500
Epoch 87/100
740/740 [==============================] - 0s 275us/sample - loss: 0.3400 -
accuracy: 0.8622
Epoch 88/100
740/740 [==============================] - 0s 276us/sample - loss: 0.3387 -
accuracy: 0.8581
Epoch 89/100
740/740 [==============================] - 0s 272us/sample - loss: 0.3365 -
accuracy: 0.8649
Epoch 90/100
740/740 [==============================] - 0s 272us/sample - loss: 0.3174 -
accuracy: 0.8689
Epoch 91/100
740/740 [==============================] - 0s 270us/sample - loss: 0.3240 -
accuracy: 0.8757
Epoch 92/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3059 -
accuracy: 0.8743
Epoch 93/100
740/740 [==============================] - 0s 272us/sample - loss: 0.3177 -
accuracy: 0.8716
Epoch 94/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3145 -
accuracy: 0.8716
Epoch 95/100
740/740 [==============================] - 0s 270us/sample - loss: 0.3313 -
accuracy: 0.8716
Epoch 96/100
740/740 [==============================] - 0s 272us/sample - loss: 0.3142 -
accuracy: 0.8703
Epoch 97/100
740/740 [==============================] - 0s 270us/sample - loss: 0.3020 -
accuracy: 0.8716
Epoch 98/100
740/740 [==============================] - 0s 266us/sample - loss: 0.3238 -
accuracy: 0.8703
Epoch 99/100
740/740 [==============================] - 0s 268us/sample - loss: 0.3275 -
accuracy: 0.8730
Epoch 100/100
740/740 [==============================] - 0s 278us/sample - loss: 0.2835 -
accuracy: 0.8946
```

[71]:
```
pyplot.plot(history_Emb3.history['loss'])
pyplot.plot(history_Emb3.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[72]:
```
val_loss_test7, val_acc_test7 = model_Emb3.evaluate([X3_test_Cat, X3_test_Num],␣
  ↪Y3_test)
val_acc_test.append(val_acc_test7)

print('Test accuracy:', val_acc_test7)
print('Test loss:', val_loss_test7)

val_loss_train7, val_acc_train7 = model_Emb3.evaluate([X3_train_Cat,␣
  ↪X3_train_Num], Y3_train)

print('Train accuracy:', val_acc_train7)
print('Train loss:', val_loss_train7)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
365/365 [==============================] - 0s 445us/sample - loss: 2.2753 -
accuracy: 0.5288
Test accuracy: 0.5287671
```

```
Test loss: 2.275326623492045
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 129us/sample - loss: 0.2218 -
accuracy: 0.9203
Train accuracy: 0.92027026
Train loss: 0.22177158074604497
```

Combining *tf_idf* and *general quoted article* feature representations with other features in the GDPR dataframe (TF-IDF+GA+OF)

[73]:
```python
df_id_names = df_Tf_Idf.iloc[:,-2:]
df_Cat_Gen_GDPR_Art_Df_new2 = pd.concat([df_Cat_Gen_GDPR_Art_Df_new1,df_new1.
 ↪iloc[:,0:5]], axis=1).reindex(df_Cat_Gen_GDPR_Art_Df_new1.index)
df_Cat_Gen_GDPR_Art_Df_new2['Fine_binned1'] = pd.
 ↪cut(df_Cat_Gen_GDPR_Art_Df_new2['Fine'], bins=cut_bins, labels=False)
df_Cat_Gen_GDPR_Art_Df_new2['Fine_binned1'] =␣
 ↪df_Cat_Gen_GDPR_Art_Df_new2['Fine_binned1'].astype(np.int64)

cols = df_Cat_Gen_GDPR_Art_Df_new2.columns.tolist()
cols = cols[:-8] + cols[-5:]+cols[len(cols)-8:len(cols)-5]

df_Cat_Gen_GDPR_Art_Df_new2 = df_Cat_Gen_GDPR_Art_Df_new2[cols]
```

[74]:
```python
# tail of the dataset
df_Cat_Gen_GDPR_Art_Df_new2.tail(5)
```

[74]:
```
      GDPR13  GDPR5  GDPR14  GDPR6  GDPR15  GDPR32  GDPR28  GDPR33  GDPR34  \
1100       0      0       0      0       0       0       0       0       0
1101       0      1       0      0       0       0       0       0       0
1102       0      0       0      0       0       0       0       0       0
1103       0      0       0      0       0       0       0       0       0
1104       0      1       0      1       0       0       0       0       0

      GDPR12     …        violat         0         1         2         3  \
1100       0     …      0.000000  0.877629 -1.360527 -1.369715  0.778230
1101       0     …      0.000000  0.877629 -0.166206  1.085728  0.778230
1102       0     …      0.424414  0.634850  0.020027 -2.842980  0.642421
1103       0     …      0.000000 -1.792937  0.015978 -2.842980 -2.209573
1104       0     …      0.000000  0.877629 -1.388867  0.103551  0.778230

             4   Fine         Id         Id  Fine_binned1
1100 -0.762759   4200  ETid-1143  ETid-1143             0
1101 -0.762759  16000  ETid-1144  ETid-1144             1
1102  0.049234   4000  ETid-1145  ETid-1145             0
1103 -0.762759  10000  ETid-1146  ETid-1146             1
1104 -1.574752   1500  ETid-1147  ETid-1147             0
```

```
[5 rows x 102 columns]
```

[75]:
```
X4 = df_Cat_Gen_GDPR_Art_Df_new2.iloc[:,0:len(df_Cat_Gen_GDPR_Art_Df_new2.
↪columns)-4]
Y4 = df_Cat_Gen_GDPR_Art_Df_new2['Fine_binned1'] #.iloc[:, -1]
X4_train, X4_test, Y4_train, Y4_test = train_test_split(X4, Y4, test_size=0.33,␣
↪random_state=42)

X4_train_Cat= X4_train.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]
X4_test_Cat= X4_test.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]

X4_train_Num= X4_train.iloc[:,len(df_Cat_Gen_GDPR_Article_Df.columns)+1:len(X4.
↪columns)]
X4_test_Num= X4_test.iloc[:,len(df_Cat_Gen_GDPR_Article_Df.columns)+1:len(X4.
↪columns)]
```

[76]:
```
# Use Input layers, specify input shape
no_of_unique_cat_gen= categorical_gen_article_df.shape[1]-1
inp_num_data = keras.layers.Input(shape=(X.shape[1]+5,))

# Creating the model
model_Emb4 = model_emb_def(no_of_unique_cat_gen,inp_num_data)
model_Emb4.summary()
```

```
Model: "model_3"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_8 (InputLayer)            [(None, 62)]         0

________________________________________________________________________________
__________________
embedding_3 (Embedding)         (None, 62, 31)       1922        input_8[0][0]

________________________________________________________________________________
__________________
flatten_3 (Flatten)             (None, 1922)         0
embedding_3[0][0]
________________________________________________________________________________
__________________
input_7 (InputLayer)            [(None, 34)]         0

________________________________________________________________________________
__________________
concatenate_3 (Concatenate)     (None, 1956)         0           flatten_3[0][0]
                                                                 input_7[0][0]
________________________________________________________________________________
```

```
------------------
dense_15 (Dense)                (None, 50)           97850
concatenate_3[0][0]
________________________________________________________________________________
------------------
dense_16 (Dense)                (None, 25)           1275        dense_15[0][0]
________________________________________________________________________________
------------------
dropout_6 (Dropout)             (None, 25)           0           dense_16[0][0]
________________________________________________________________________________
------------------
dense_17 (Dense)                (None, 15)           390         dropout_6[0][0]
________________________________________________________________________________
------------------
dropout_7 (Dropout)             (None, 15)           0           dense_17[0][0]
________________________________________________________________________________
------------------
dense_18 (Dense)                (None, 10)           160         dropout_7[0][0]
________________________________________________________________________________
------------------
dense_19 (Dense)                (None, 5)            55          dense_18[0][0]
================================================================================
==================
Total params: 101,652
Trainable params: 101,652
Non-trainable params: 0
________________________________________________________________________________
------------------
```

[77]:
```
model_Emb4.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
  ↪metrics=['accuracy'])
history_Emb4 = model_Emb4.fit([X4_train_Cat, X4_train_Num],␣
  ↪Y4_train,epochs=100, batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 732us/sample - loss: 1.5939 -
accuracy: 0.3568
Epoch 2/100
740/740 [==============================] - 0s 135us/sample - loss: 1.5103 -
accuracy: 0.4324
Epoch 3/100
740/740 [==============================] - 0s 142us/sample - loss: 1.4050 -
accuracy: 0.4459
Epoch 4/100
```

```
740/740 [==============================] - 0s 137us/sample - loss: 1.2883 -
accuracy: 0.4568
Epoch 5/100
740/740 [==============================] - 0s 140us/sample - loss: 1.2174 -
accuracy: 0.4459
Epoch 6/100
740/740 [==============================] - 0s 168us/sample - loss: 1.1658 -
accuracy: 0.4622
Epoch 7/100
740/740 [==============================] - 0s 151us/sample - loss: 1.1362 -
accuracy: 0.4486
Epoch 8/100
740/740 [==============================] - 0s 139us/sample - loss: 1.1210 -
accuracy: 0.4500
Epoch 9/100
740/740 [==============================] - 0s 146us/sample - loss: 1.0993 -
accuracy: 0.4446
Epoch 10/100
740/740 [==============================] - 0s 144us/sample - loss: 1.0974 -
accuracy: 0.4703
Epoch 11/100
740/740 [==============================] - 0s 187us/sample - loss: 1.0845 -
accuracy: 0.4486
Epoch 12/100
740/740 [==============================] - 0s 162us/sample - loss: 1.0643 -
accuracy: 0.4689
Epoch 13/100
740/740 [==============================] - 0s 146us/sample - loss: 1.0704 -
accuracy: 0.5027
Epoch 14/100
740/740 [==============================] - 0s 142us/sample - loss: 1.0692 -
accuracy: 0.4527
Epoch 15/100
740/740 [==============================] - 0s 136us/sample - loss: 1.0416 -
accuracy: 0.5095
Epoch 16/100
740/740 [==============================] - 0s 150us/sample - loss: 1.0242 -
accuracy: 0.5189
Epoch 17/100
740/740 [==============================] - 0s 158us/sample - loss: 1.0197 -
accuracy: 0.4986
Epoch 18/100
740/740 [==============================] - 0s 140us/sample - loss: 1.0342 -
accuracy: 0.5324
Epoch 19/100
740/740 [==============================] - 0s 137us/sample - loss: 1.0173 -
accuracy: 0.5203
Epoch 20/100
```

```
740/740 [==============================] - 0s 139us/sample - loss: 0.9972 -
accuracy: 0.5541
Epoch 21/100
740/740 [==============================] - 0s 155us/sample - loss: 0.9924 -
accuracy: 0.5568
Epoch 22/100
740/740 [==============================] - 0s 155us/sample - loss: 0.9734 -
accuracy: 0.5419
Epoch 23/100
740/740 [==============================] - 0s 147us/sample - loss: 0.9609 -
accuracy: 0.5581
Epoch 24/100
740/740 [==============================] - 0s 140us/sample - loss: 0.9673 -
accuracy: 0.5878
Epoch 25/100
740/740 [==============================] - 0s 158us/sample - loss: 0.9454 -
accuracy: 0.5730
Epoch 26/100
740/740 [==============================] - 0s 147us/sample - loss: 0.9231 -
accuracy: 0.5878
Epoch 27/100
740/740 [==============================] - 0s 136us/sample - loss: 0.9118 -
accuracy: 0.5959
Epoch 28/100
740/740 [==============================] - 0s 139us/sample - loss: 0.8777 -
accuracy: 0.6243
Epoch 29/100
740/740 [==============================] - 0s 137us/sample - loss: 0.8948 -
accuracy: 0.6014
Epoch 30/100
740/740 [==============================] - 0s 143us/sample - loss: 0.8446 -
accuracy: 0.6135
Epoch 31/100
740/740 [==============================] - 0s 137us/sample - loss: 0.8166 -
accuracy: 0.6608
Epoch 32/100
740/740 [==============================] - 0s 136us/sample - loss: 0.8285 -
accuracy: 0.6378
Epoch 33/100
740/740 [==============================] - 0s 137us/sample - loss: 0.8035 -
accuracy: 0.6527
Epoch 34/100
740/740 [==============================] - 0s 137us/sample - loss: 0.8004 -
accuracy: 0.6486
Epoch 35/100
740/740 [==============================] - 0s 142us/sample - loss: 0.7802 -
accuracy: 0.6689
Epoch 36/100
```

```
740/740 [==============================] - 0s 142us/sample - loss: 0.7621 -
accuracy: 0.6622
Epoch 37/100
740/740 [==============================] - 0s 139us/sample - loss: 0.7527 -
accuracy: 0.6824
Epoch 38/100
740/740 [==============================] - 0s 142us/sample - loss: 0.7450 -
accuracy: 0.6797
Epoch 39/100
740/740 [==============================] - 0s 139us/sample - loss: 0.7347 -
accuracy: 0.6838
Epoch 40/100
740/740 [==============================] - 0s 140us/sample - loss: 0.7195 -
accuracy: 0.6946
Epoch 41/100
740/740 [==============================] - 0s 136us/sample - loss: 0.6737 -
accuracy: 0.7203
Epoch 42/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6820 -
accuracy: 0.7014
Epoch 43/100
740/740 [==============================] - 0s 140us/sample - loss: 0.6492 -
accuracy: 0.7324
Epoch 44/100
740/740 [==============================] - 0s 135us/sample - loss: 0.6621 -
accuracy: 0.7203
Epoch 45/100
740/740 [==============================] - 0s 142us/sample - loss: 0.6599 -
accuracy: 0.7338
Epoch 46/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6242 -
accuracy: 0.7527
Epoch 47/100
740/740 [==============================] - 0s 136us/sample - loss: 0.6198 -
accuracy: 0.7338
Epoch 48/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6323 -
accuracy: 0.7338
Epoch 49/100
740/740 [==============================] - 0s 137us/sample - loss: 0.6123 -
accuracy: 0.7405
Epoch 50/100
740/740 [==============================] - 0s 136us/sample - loss: 0.5966 -
accuracy: 0.7635
Epoch 51/100
740/740 [==============================] - 0s 139us/sample - loss: 0.6065 -
accuracy: 0.7432
Epoch 52/100
```

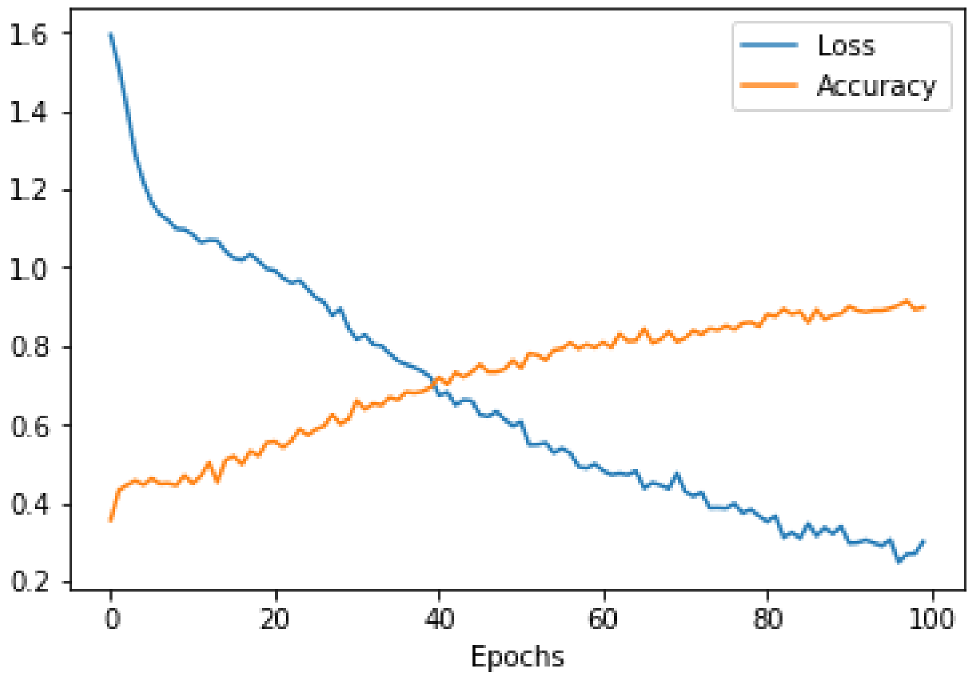
```
740/740 [==============================] - 0s 140us/sample - loss: 0.5482 -
accuracy: 0.7811
Epoch 53/100
740/740 [==============================] - 0s 136us/sample - loss: 0.5487 -
accuracy: 0.7770
Epoch 54/100
740/740 [==============================] - 0s 139us/sample - loss: 0.5537 -
accuracy: 0.7635
Epoch 55/100
740/740 [==============================] - 0s 140us/sample - loss: 0.5279 -
accuracy: 0.7892
Epoch 56/100
740/740 [==============================] - 0s 140us/sample - loss: 0.5390 -
accuracy: 0.7932
Epoch 57/100
740/740 [==============================] - 0s 140us/sample - loss: 0.5273 -
accuracy: 0.8081
Epoch 58/100
740/740 [==============================] - 0s 135us/sample - loss: 0.4924 -
accuracy: 0.7932
Epoch 59/100
740/740 [==============================] - 0s 139us/sample - loss: 0.4886 -
accuracy: 0.8041
Epoch 60/100
740/740 [==============================] - 0s 136us/sample - loss: 0.4993 -
accuracy: 0.7959
Epoch 61/100
740/740 [==============================] - 0s 137us/sample - loss: 0.4821 -
accuracy: 0.8095
Epoch 62/100
740/740 [==============================] - 0s 142us/sample - loss: 0.4717 -
accuracy: 0.7959
Epoch 63/100
740/740 [==============================] - 0s 135us/sample - loss: 0.4754 -
accuracy: 0.8311
Epoch 64/100
740/740 [==============================] - 0s 139us/sample - loss: 0.4716 -
accuracy: 0.8122
Epoch 65/100
740/740 [==============================] - 0s 143us/sample - loss: 0.4813 -
accuracy: 0.8135
Epoch 66/100
740/740 [==============================] - 0s 139us/sample - loss: 0.4371 -
accuracy: 0.8446
Epoch 67/100
740/740 [==============================] - 0s 139us/sample - loss: 0.4516 -
accuracy: 0.8081
Epoch 68/100
```

```
740/740 [==============================] - 0s 136us/sample - loss: 0.4452 -
accuracy: 0.8162
Epoch 69/100
740/740 [==============================] - 0s 136us/sample - loss: 0.4361 -
accuracy: 0.8365
Epoch 70/100
740/740 [==============================] - 0s 143us/sample - loss: 0.4757 -
accuracy: 0.8108
Epoch 71/100
740/740 [==============================] - 0s 160us/sample - loss: 0.4288 -
accuracy: 0.8203
Epoch 72/100
740/740 [==============================] - 0s 160us/sample - loss: 0.4170 -
accuracy: 0.8392
Epoch 73/100
740/740 [==============================] - 0s 146us/sample - loss: 0.4268 -
accuracy: 0.8297
Epoch 74/100
740/740 [==============================] - 0s 147us/sample - loss: 0.3871 -
accuracy: 0.8446
Epoch 75/100
740/740 [==============================] - 0s 139us/sample - loss: 0.3877 -
accuracy: 0.8405
Epoch 76/100
740/740 [==============================] - 0s 139us/sample - loss: 0.3860 -
accuracy: 0.8514
Epoch 77/100
740/740 [==============================] - 0s 137us/sample - loss: 0.3978 -
accuracy: 0.8419
Epoch 78/100
740/740 [==============================] - 0s 137us/sample - loss: 0.3740 -
accuracy: 0.8581
Epoch 79/100
740/740 [==============================] - 0s 142us/sample - loss: 0.3832 -
accuracy: 0.8608
Epoch 80/100
740/740 [==============================] - 0s 136us/sample - loss: 0.3660 -
accuracy: 0.8500
Epoch 81/100
740/740 [==============================] - 0s 139us/sample - loss: 0.3523 -
accuracy: 0.8811
Epoch 82/100
740/740 [==============================] - 0s 135us/sample - loss: 0.3665 -
accuracy: 0.8757
Epoch 83/100
740/740 [==============================] - 0s 136us/sample - loss: 0.3114 -
accuracy: 0.8946
Epoch 84/100
```

```
740/740 [==============================] - 0s 137us/sample - loss: 0.3246 -
accuracy: 0.8811
Epoch 85/100
740/740 [==============================] - 0s 142us/sample - loss: 0.3095 -
accuracy: 0.8878
Epoch 86/100
740/740 [==============================] - 0s 181us/sample - loss: 0.3475 -
accuracy: 0.8595
Epoch 87/100
740/740 [==============================] - 0s 143us/sample - loss: 0.3152 -
accuracy: 0.8932
Epoch 88/100
740/740 [==============================] - 0s 136us/sample - loss: 0.3369 -
accuracy: 0.8676
Epoch 89/100
740/740 [==============================] - 0s 136us/sample - loss: 0.3203 -
accuracy: 0.8784
Epoch 90/100
740/740 [==============================] - 0s 137us/sample - loss: 0.3399 -
accuracy: 0.8824
Epoch 91/100
740/740 [==============================] - 0s 137us/sample - loss: 0.2969 -
accuracy: 0.9014
Epoch 92/100
740/740 [==============================] - 0s 139us/sample - loss: 0.2982 -
accuracy: 0.8905
Epoch 93/100
740/740 [==============================] - 0s 140us/sample - loss: 0.3054 -
accuracy: 0.8865
Epoch 94/100
740/740 [==============================] - 0s 139us/sample - loss: 0.2973 -
accuracy: 0.8905
Epoch 95/100
740/740 [==============================] - 0s 136us/sample - loss: 0.2900 -
accuracy: 0.8905
Epoch 96/100
740/740 [==============================] - 0s 136us/sample - loss: 0.3064 -
accuracy: 0.8959
Epoch 97/100
740/740 [==============================] - 0s 135us/sample - loss: 0.2482 -
accuracy: 0.9041
Epoch 98/100
740/740 [==============================] - 0s 137us/sample - loss: 0.2688 -
accuracy: 0.9149
Epoch 99/100
740/740 [==============================] - 0s 137us/sample - loss: 0.2705 -
accuracy: 0.8932
Epoch 100/100
```

740/740 [==============================] - 0s 136us/sample - loss: 0.3008 - accuracy: 0.8986

[78]:
```
pyplot.plot(history_Emb4.history['loss'])
pyplot.plot(history_Emb4.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[79]:
```
val_loss_test8, val_acc_test8 = model_Emb4.evaluate([X4_test_Cat, X4_test_Num],␣
 ↪Y4_test)
val_acc_test.append(val_acc_test8)

print('Test accuracy:', val_acc_test8)
print('Test loss:', val_loss_test8)

val_loss_train8, val_acc_train8 = model_Emb4.evaluate([X4_train_Cat,␣
 ↪X4_train_Num], Y4_train)

print('Train accuracy:', val_acc_train8)
print('Train loss:', val_loss_train8)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
```

```
365/365 [==============================] - 0s 399us/sample - loss: 2.1335 -
accuracy: 0.5452
Test accuracy: 0.5452055
Test loss: 2.133533140077983
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 81us/sample - loss: 0.2122 -
accuracy: 0.9284
Train accuracy: 0.9283784
Train loss: 0.21222980152312163
```

Analysis based on other features in the GDPR dataframe (OF)

[80]:
```python
X_OF = df_new1.iloc[:,0:5]
Y_OF = df_new1['Fine_binned1']

X_train_OF, X_test_OF, Y_train_OF, Y_test_OF = train_test_split(X_OF, Y_OF,
 ↪test_size=0.33, random_state=42)
```

[81]:
```python
model_OF = Sequential()
model_OF.add(Dense(units=50, activation='relu', input_dim=(5),
 ↪kernel_constraint=unit_norm()))
model_OF.add(Dropout(.2))
model_OF.add(Dense(units=15, activation='relu'))
model_OF.add(Dropout(.2))
model_OF.add(Dense(units=10, activation='relu'))
model_OF.add(Dense(units=5, activation='softmax'))
model_OF.compile(loss='sparse_categorical_crossentropy', optimizer="adam",
 ↪metrics=['accuracy'])
model_OF.summary()

history_OF=model_OF.fit(X_train_OF, Y_train_OF, epochs=100, batch_size=32)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                300
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 15)                765
_________________________________________________________________
dropout_2 (Dropout)          (None, 15)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                160
_________________________________________________________________
```

```
dense_4 (Dense)              (None, 5)                 55
=================================================================
Total params: 1,280
Trainable params: 1,280
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
740/740 [==============================] - 1s 2ms/step - loss: 1.6007 -
accuracy: 0.3257
Epoch 2/100
740/740 [==============================] - 0s 139us/step - loss: 1.3597 -
accuracy: 0.4351
Epoch 3/100
740/740 [==============================] - 0s 143us/step - loss: 1.2407 -
accuracy: 0.4649
Epoch 4/100
740/740 [==============================] - 0s 113us/step - loss: 1.2070 -
accuracy: 0.4595
Epoch 5/100
740/740 [==============================] - 0s 111us/step - loss: 1.1593 -
accuracy: 0.4784
Epoch 6/100
740/740 [==============================] - 0s 96us/step - loss: 1.1206 -
accuracy: 0.5027
Epoch 7/100
740/740 [==============================] - 0s 113us/step - loss: 1.1324 -
accuracy: 0.5054
Epoch 8/100
740/740 [==============================] - 0s 116us/step - loss: 1.1330 -
accuracy: 0.4730
Epoch 9/100
740/740 [==============================] - 0s 102us/step - loss: 1.1229 -
accuracy: 0.5000
Epoch 10/100
740/740 [==============================] - 0s 117us/step - loss: 1.1034 -
accuracy: 0.4730
Epoch 11/100
740/740 [==============================] - 0s 92us/step - loss: 1.1127 -
accuracy: 0.4770
Epoch 12/100
740/740 [==============================] - 0s 132us/step - loss: 1.1002 -
accuracy: 0.5054
Epoch 13/100
740/740 [==============================] - 0s 109us/step - loss: 1.0772 -
accuracy: 0.5054
Epoch 14/100
740/740 [==============================] - 0s 106us/step - loss: 1.1004 -
accuracy: 0.4622
```

```
Epoch 15/100
740/740 [==============================] - 0s 142us/step - loss: 1.0910 -
accuracy: 0.5027
Epoch 16/100
740/740 [==============================] - 0s 104us/step - loss: 1.0776 -
accuracy: 0.5081
Epoch 17/100
740/740 [==============================] - 0s 105us/step - loss: 1.0609 -
accuracy: 0.4973
Epoch 18/100
740/740 [==============================] - 0s 96us/step - loss: 1.0817 -
accuracy: 0.4797
Epoch 19/100
740/740 [==============================] - 0s 129us/step - loss: 1.0762 -
accuracy: 0.5351
Epoch 20/100
740/740 [==============================] - 0s 114us/step - loss: 1.0621 -
accuracy: 0.5027
Epoch 21/100
740/740 [==============================] - 0s 102us/step - loss: 1.0684 -
accuracy: 0.5135
Epoch 22/100
740/740 [==============================] - 0s 112us/step - loss: 1.0745 -
accuracy: 0.4838
Epoch 23/100
740/740 [==============================] - 0s 102us/step - loss: 1.0610 -
accuracy: 0.5189
Epoch 24/100
740/740 [==============================] - 0s 98us/step - loss: 1.0503 -
accuracy: 0.5473
Epoch 25/100
740/740 [==============================] - 0s 144us/step - loss: 1.0501 -
accuracy: 0.5149
Epoch 26/100
740/740 [==============================] - 0s 123us/step - loss: 1.0552 -
accuracy: 0.5324
Epoch 27/100
740/740 [==============================] - 0s 111us/step - loss: 1.0530 -
accuracy: 0.5473
Epoch 28/100
740/740 [==============================] - 0s 96us/step - loss: 1.0302 -
accuracy: 0.5473
Epoch 29/100
740/740 [==============================] - 0s 99us/step - loss: 1.0519 -
accuracy: 0.5135
Epoch 30/100
740/740 [==============================] - 0s 97us/step - loss: 1.0484 -
accuracy: 0.5216
```

```
Epoch 31/100
740/740 [==============================] - 0s 94us/step - loss: 1.0482 -
accuracy: 0.4946
Epoch 32/100
740/740 [==============================] - 0s 115us/step - loss: 1.0462 -
accuracy: 0.5595
Epoch 33/100
740/740 [==============================] - 0s 102us/step - loss: 1.0454 -
accuracy: 0.5095
Epoch 34/100
740/740 [==============================] - 0s 97us/step - loss: 1.0474 -
accuracy: 0.5216
Epoch 35/100
740/740 [==============================] - 0s 96us/step - loss: 1.0540 -
accuracy: 0.5257
Epoch 36/100
740/740 [==============================] - 0s 92us/step - loss: 1.0557 -
accuracy: 0.5149
Epoch 37/100
740/740 [==============================] - 0s 96us/step - loss: 1.0363 -
accuracy: 0.5297
Epoch 38/100
740/740 [==============================] - 0s 96us/step - loss: 1.0351 -
accuracy: 0.5392
Epoch 39/100
740/740 [==============================] - 0s 105us/step - loss: 1.0150 -
accuracy: 0.5486
Epoch 40/100
740/740 [==============================] - 0s 129us/step - loss: 1.0498 -
accuracy: 0.5230
Epoch 41/100
740/740 [==============================] - 0s 98us/step - loss: 1.0528 -
accuracy: 0.5149
Epoch 42/100
740/740 [==============================] - 0s 134us/step - loss: 1.0338 -
accuracy: 0.5297
Epoch 43/100
740/740 [==============================] - 0s 123us/step - loss: 1.0382 -
accuracy: 0.5527
Epoch 44/100
740/740 [==============================] - 0s 116us/step - loss: 1.0283 -
accuracy: 0.5365
Epoch 45/100
740/740 [==============================] - 0s 133us/step - loss: 1.0263 -
accuracy: 0.5257
Epoch 46/100
740/740 [==============================] - 0s 116us/step - loss: 1.0343 -
accuracy: 0.5365
```

```
Epoch 47/100
740/740 [==============================] - 0s 100us/step - loss: 1.0312 -
accuracy: 0.5432
Epoch 48/100
740/740 [==============================] - 0s 128us/step - loss: 1.0387 -
accuracy: 0.5108
Epoch 49/100
740/740 [==============================] - 0s 112us/step - loss: 1.0294 -
accuracy: 0.5270
Epoch 50/100
740/740 [==============================] - 0s 99us/step - loss: 1.0346 -
accuracy: 0.5095
Epoch 51/100
740/740 [==============================] - 0s 102us/step - loss: 1.0309 -
accuracy: 0.5365
Epoch 52/100
740/740 [==============================] - 0s 131us/step - loss: 1.0318 -
accuracy: 0.5189
Epoch 53/100
740/740 [==============================] - 0s 107us/step - loss: 1.0345 -
accuracy: 0.5338
Epoch 54/100
740/740 [==============================] - 0s 108us/step - loss: 1.0132 -
accuracy: 0.5581
Epoch 55/100
740/740 [==============================] - 0s 106us/step - loss: 1.0236 -
accuracy: 0.5676
Epoch 56/100
740/740 [==============================] - 0s 112us/step - loss: 1.0125 -
accuracy: 0.5459
Epoch 57/100
740/740 [==============================] - 0s 107us/step - loss: 1.0185 -
accuracy: 0.5311
Epoch 58/100
740/740 [==============================] - 0s 88us/step - loss: 1.0146 -
accuracy: 0.5473
Epoch 59/100
740/740 [==============================] - 0s 94us/step - loss: 1.0202 -
accuracy: 0.5419
Epoch 60/100
740/740 [==============================] - 0s 91us/step - loss: 1.0195 -
accuracy: 0.5608
Epoch 61/100
740/740 [==============================] - 0s 97us/step - loss: 1.0211 -
accuracy: 0.5514
Epoch 62/100
740/740 [==============================] - 0s 106us/step - loss: 1.0176 -
accuracy: 0.5432
```

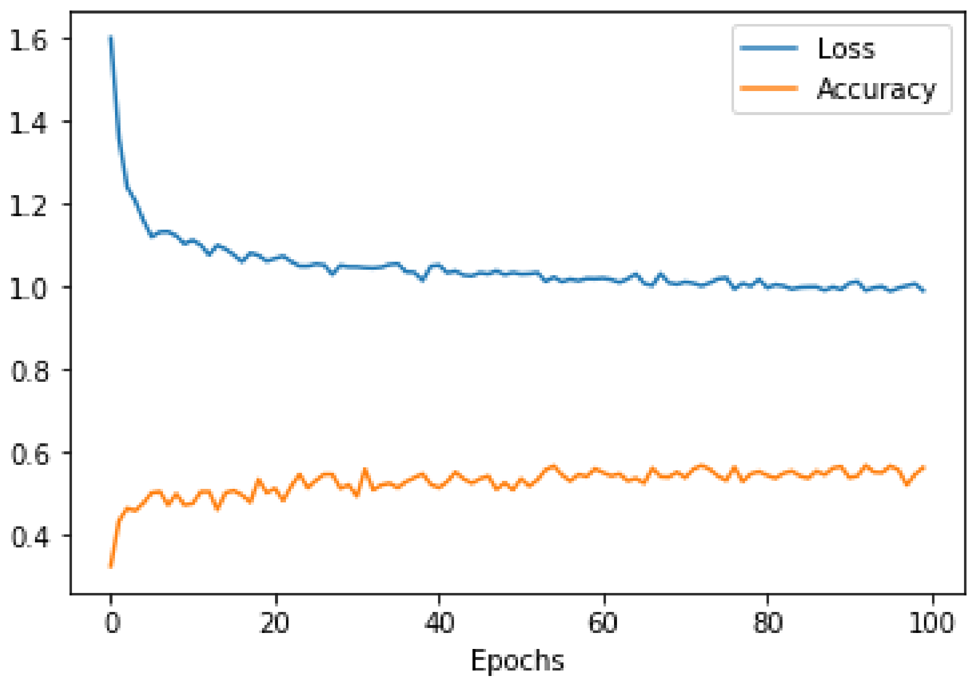
```
Epoch 63/100
740/740 [==============================] - 0s 133us/step - loss: 1.0106 -
accuracy: 0.5486
Epoch 64/100
740/740 [==============================] - 0s 109us/step - loss: 1.0195 -
accuracy: 0.5311
Epoch 65/100
740/740 [==============================] - 0s 103us/step - loss: 1.0316 -
accuracy: 0.5378
Epoch 66/100
740/740 [==============================] - 0s 86us/step - loss: 1.0082 -
accuracy: 0.5257
Epoch 67/100
740/740 [==============================] - 0s 96us/step - loss: 1.0031 -
accuracy: 0.5622
Epoch 68/100
740/740 [==============================] - 0s 92us/step - loss: 1.0323 -
accuracy: 0.5419
Epoch 69/100
740/740 [==============================] - 0s 128us/step - loss: 1.0099 -
accuracy: 0.5405
Epoch 70/100
740/740 [==============================] - 0s 126us/step - loss: 1.0062 -
accuracy: 0.5527
Epoch 71/100
740/740 [==============================] - 0s 108us/step - loss: 1.0121 -
accuracy: 0.5392
Epoch 72/100
740/740 [==============================] - 0s 111us/step - loss: 1.0088 -
accuracy: 0.5595
Epoch 73/100
740/740 [==============================] - 0s 96us/step - loss: 1.0024 -
accuracy: 0.5689
Epoch 74/100
740/740 [==============================] - 0s 101us/step - loss: 1.0095 -
accuracy: 0.5595
Epoch 75/100
740/740 [==============================] - 0s 105us/step - loss: 1.0192 -
accuracy: 0.5432
Epoch 76/100
740/740 [==============================] - 0s 131us/step - loss: 1.0223 -
accuracy: 0.5324
Epoch 77/100
740/740 [==============================] - 0s 110us/step - loss: 0.9949 -
accuracy: 0.5649
Epoch 78/100
740/740 [==============================] - 0s 110us/step - loss: 1.0085 -
accuracy: 0.5297
```

```
Epoch 79/100
740/740 [==============================] - 0s 129us/step - loss: 1.0015 -
accuracy: 0.5486
Epoch 80/100
740/740 [==============================] - 0s 133us/step - loss: 1.0182 -
accuracy: 0.5541
Epoch 81/100
740/740 [==============================] - 0s 117us/step - loss: 0.9978 -
accuracy: 0.5446
Epoch 82/100
740/740 [==============================] - 0s 106us/step - loss: 1.0054 -
accuracy: 0.5378
Epoch 83/100
740/740 [==============================] - 0s 103us/step - loss: 1.0021 -
accuracy: 0.5500
Epoch 84/100
740/740 [==============================] - 0s 100us/step - loss: 0.9946 -
accuracy: 0.5554
Epoch 85/100
740/740 [==============================] - 0s 109us/step - loss: 0.9988 -
accuracy: 0.5432
Epoch 86/100
740/740 [==============================] - 0s 111us/step - loss: 0.9999 -
accuracy: 0.5378
Epoch 87/100
740/740 [==============================] - 0s 97us/step - loss: 1.0001 -
accuracy: 0.5554
Epoch 88/100
740/740 [==============================] - 0s 115us/step - loss: 0.9914 -
accuracy: 0.5459
Epoch 89/100
740/740 [==============================] - 0s 105us/step - loss: 0.9999 -
accuracy: 0.5622
Epoch 90/100
740/740 [==============================] - 0s 121us/step - loss: 0.9934 -
accuracy: 0.5662
Epoch 91/100
740/740 [==============================] - 0s 133us/step - loss: 1.0088 -
accuracy: 0.5392
Epoch 92/100
740/740 [==============================] - 0s 134us/step - loss: 1.0132 -
accuracy: 0.5432
Epoch 93/100
740/740 [==============================] - 0s 129us/step - loss: 0.9910 -
accuracy: 0.5689
Epoch 94/100
740/740 [==============================] - 0s 138us/step - loss: 0.9979 -
accuracy: 0.5527
```

```
Epoch 95/100
740/740 [==============================] - 0s 137us/step - loss: 1.0008 -
accuracy: 0.5514
Epoch 96/100
740/740 [==============================] - 0s 129us/step - loss: 0.9895 -
accuracy: 0.5676
Epoch 97/100
740/740 [==============================] - 0s 126us/step - loss: 0.9972 -
accuracy: 0.5581
Epoch 98/100
740/740 [==============================] - 0s 129us/step - loss: 1.0027 -
accuracy: 0.5216
Epoch 99/100
740/740 [==============================] - 0s 138us/step - loss: 1.0074 -
accuracy: 0.5473
Epoch 100/100
740/740 [==============================] - 0s 108us/step - loss: 0.9908 -
accuracy: 0.5635
```

[82]:

```
pyplot.plot(history_OF.history['loss'])
pyplot.plot(history_OF.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[83]:
```
val_loss_test9, val_acc_test9 = model_OF.evaluate(X_test_OF, Y_test_OF)
val_acc_test.append(val_acc_test9)

print('Test accuracy:', val_acc_test9)
print('Test loss:', val_loss_test9)

val_loss_train9, val_acc_train9 = model_OF.evaluate(X_train_OF, Y_train_OF)

print('Train accuracy:', val_acc_train9)
print('Train loss:', val_loss_train9)
```

```
365/365 [==============================] - 0s 713us/step
Test accuracy: 0.567123293876648
Test loss: 1.0343643900466293
740/740 [==============================] - 0s 50us/step
Train accuracy: 0.5648648738861084
Train loss: 0.9538864883216651
```

Combining specific quoted article feature representations with other features in the GDPR dataframe (SA+OF)

[84]:
```
df_OF = df_new1.iloc[:,[0,1,2,3,4,len(df_new1.columns)-1]]
df_OF_SA_Df = pd.concat([df_Cat_GDPR_Article_Df, df_OF], axis=1).
↪reindex(df_Cat_GDPR_Article_Df.index)
df_OF_SA_Df = df_OF_SA_Df.drop(['Id'], axis = 1)
```

[85]:
```
X5 = df_OF_SA_Df.iloc[:,0:len(df_OF_SA_Df.columns)-1]
Y5 = df_OF_SA_Df['Fine_binned1']
X5_train, X5_test, Y5_train, Y5_test = train_test_split(X5, Y5, test_size=0.33,␣
↪random_state=42)

X5_train_Cat= X5_train.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]
X5_test_Cat= X5_test.iloc[:,0:len(df_Cat_GDPR_Article_Df.columns)-1]

X5_train_Num= X5_train.iloc[:,len(df_Cat_GDPR_Article_Df.columns)-1:len(X5.
↪columns)]
X5_test_Num= X5_test.iloc[:,len(df_Cat_GDPR_Article_Df.columns)-1:len(X5.
↪columns)]
```

[86]:
```
# Use Input layers, specify input shape
no_of_unique_cat= len(df_Cat_GDPR_Article_Df.columns)-1
inp_num_data = keras.layers.Input(shape=(5,))

# Creating the model
model_Emb5 = model_emb_def(no_of_unique_cat,inp_num_data)
model_Emb5.summary()
```

```
Model: "model"
```

```
__________________________________________________________________________________
________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
================
input_2 (InputLayer)            [(None, 244)]        0

__________________________________________________________________________________
________________
embedding (Embedding)           (None, 244, 50)      12200       input_2[0][0]

__________________________________________________________________________________
________________
flatten (Flatten)               (None, 12200)        0           embedding[0][0]

__________________________________________________________________________________
________________
input_1 (InputLayer)            [(None, 5)]          0

__________________________________________________________________________________
________________
concatenate (Concatenate)       (None, 12205)        0           flatten[0][0]
                                                                 input_1[0][0]

__________________________________________________________________________________
________________
dense (Dense)                   (None, 50)           610300
concatenate[0][0]

__________________________________________________________________________________
________________
dense_1 (Dense)                 (None, 25)           1275        dense[0][0]

__________________________________________________________________________________
________________
dropout (Dropout)               (None, 25)           0           dense_1[0][0]

__________________________________________________________________________________
________________
dense_2 (Dense)                 (None, 15)           390         dropout[0][0]

__________________________________________________________________________________
________________
dropout_1 (Dropout)             (None, 15)           0           dense_2[0][0]

__________________________________________________________________________________
________________
dense_3 (Dense)                 (None, 10)           160         dropout_1[0][0]

__________________________________________________________________________________
________________
dense_4 (Dense)                 (None, 5)            55          dense_3[0][0]
==================================================================================
================
Total params: 624,380
Trainable params: 624,380
Non-trainable params: 0

__________________________________________________________________________________
________________
```

[87]:
```
model_Emb5.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
↪metrics=['accuracy'])
history_Emb5 = model_Emb5.fit([X5_train_Cat, X5_train_Num],␣
↪Y5_train,epochs=100, batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 2ms/sample - loss: 1.4134 -
accuracy: 0.4270
Epoch 2/100
740/740 [==============================] - 0s 418us/sample - loss: 1.2828 -
accuracy: 0.4081
Epoch 3/100
740/740 [==============================] - 0s 409us/sample - loss: 1.2225 -
accuracy: 0.4432
Epoch 4/100
740/740 [==============================] - 0s 408us/sample - loss: 1.1833 -
accuracy: 0.4365
Epoch 5/100
740/740 [==============================] - 0s 411us/sample - loss: 1.2130 -
accuracy: 0.4162
Epoch 6/100
740/740 [==============================] - 0s 427us/sample - loss: 1.1458 -
accuracy: 0.4459
Epoch 7/100
740/740 [==============================] - 0s 420us/sample - loss: 1.1280 -
accuracy: 0.4608
Epoch 8/100
740/740 [==============================] - 0s 427us/sample - loss: 1.0794 -
accuracy: 0.5162
Epoch 9/100
740/740 [==============================] - 0s 428us/sample - loss: 1.0709 -
accuracy: 0.5297
Epoch 10/100
740/740 [==============================] - 0s 439us/sample - loss: 1.0474 -
accuracy: 0.5135
Epoch 11/100
740/740 [==============================] - 0s 425us/sample - loss: 1.0387 -
accuracy: 0.5622
Epoch 12/100
740/740 [==============================] - 0s 429us/sample - loss: 0.9998 -
accuracy: 0.5635
Epoch 13/100
740/740 [==============================] - 0s 434us/sample - loss: 0.9982 -
```

```
accuracy: 0.5554
Epoch 14/100
740/740 [==============================] - 0s 423us/sample - loss: 0.9647 -
accuracy: 0.5865
Epoch 15/100
740/740 [==============================] - 0s 433us/sample - loss: 0.9524 -
accuracy: 0.5986
Epoch 16/100
740/740 [==============================] - 0s 420us/sample - loss: 0.9328 -
accuracy: 0.6014
Epoch 17/100
740/740 [==============================] - 0s 422us/sample - loss: 0.9239 -
accuracy: 0.6122
Epoch 18/100
740/740 [==============================] - 0s 420us/sample - loss: 0.9059 -
accuracy: 0.6081
Epoch 19/100
740/740 [==============================] - 0s 422us/sample - loss: 0.9113 -
accuracy: 0.6081
Epoch 20/100
740/740 [==============================] - 0s 414us/sample - loss: 0.8892 -
accuracy: 0.6338
Epoch 21/100
740/740 [==============================] - 0s 425us/sample - loss: 0.8868 -
accuracy: 0.6351
Epoch 22/100
740/740 [==============================] - 0s 432us/sample - loss: 0.8534 -
accuracy: 0.6351
Epoch 23/100
740/740 [==============================] - 0s 422us/sample - loss: 0.8432 -
accuracy: 0.6324
Epoch 24/100
740/740 [==============================] - 0s 422us/sample - loss: 0.8372 -
accuracy: 0.6392
Epoch 25/100
740/740 [==============================] - 0s 389us/sample - loss: 0.8110 -
accuracy: 0.6378
Epoch 26/100
740/740 [==============================] - 0s 441us/sample - loss: 0.8216 -
accuracy: 0.6635
Epoch 27/100
740/740 [==============================] - 0s 424us/sample - loss: 0.8201 -
accuracy: 0.6500
Epoch 28/100
740/740 [==============================] - 0s 410us/sample - loss: 0.7939 -
accuracy: 0.6595
Epoch 29/100
740/740 [==============================] - 0s 419us/sample - loss: 0.7682 -
```

```
accuracy: 0.6730
Epoch 30/100
740/740 [==============================] - 0s 409us/sample - loss: 0.7741 -
accuracy: 0.6662
Epoch 31/100
740/740 [==============================] - 0s 481us/sample - loss: 0.7768 -
accuracy: 0.6689
Epoch 32/100
740/740 [==============================] - 0s 413us/sample - loss: 0.7823 -
accuracy: 0.6757
Epoch 33/100
740/740 [==============================] - 0s 398us/sample - loss: 0.7734 -
accuracy: 0.6865
Epoch 34/100
740/740 [==============================] - 0s 381us/sample - loss: 0.7365 -
accuracy: 0.6865
Epoch 35/100
740/740 [==============================] - 0s 414us/sample - loss: 0.7394 -
accuracy: 0.7054
Epoch 36/100
740/740 [==============================] - 0s 398us/sample - loss: 0.7670 -
accuracy: 0.6770
Epoch 37/100
740/740 [==============================] - 0s 435us/sample - loss: 0.7241 -
accuracy: 0.7068
Epoch 38/100
740/740 [==============================] - 0s 419us/sample - loss: 0.6790 -
accuracy: 0.7176
Epoch 39/100
740/740 [==============================] - 0s 410us/sample - loss: 0.7052 -
accuracy: 0.7014
Epoch 40/100
740/740 [==============================] - 0s 425us/sample - loss: 0.7063 -
accuracy: 0.7189
Epoch 41/100
740/740 [==============================] - 0s 423us/sample - loss: 0.6865 -
accuracy: 0.7149
Epoch 42/100
740/740 [==============================] - 0s 415us/sample - loss: 0.7138 -
accuracy: 0.6986
Epoch 43/100
740/740 [==============================] - 0s 410us/sample - loss: 0.7025 -
accuracy: 0.7081
Epoch 44/100
740/740 [==============================] - 0s 414us/sample - loss: 0.6739 -
accuracy: 0.7284
Epoch 45/100
740/740 [==============================] - 0s 411us/sample - loss: 0.6672 -
```

```
accuracy: 0.7189
Epoch 46/100
740/740 [==============================] - 0s 430us/sample - loss: 0.6697 -
accuracy: 0.7189
Epoch 47/100
740/740 [==============================] - 0s 424us/sample - loss: 0.6667 -
accuracy: 0.7203
Epoch 48/100
740/740 [==============================] - 0s 412us/sample - loss: 0.6443 -
accuracy: 0.7311
Epoch 49/100
740/740 [==============================] - 1s 744us/sample - loss: 0.6289 -
accuracy: 0.7419
Epoch 50/100
740/740 [==============================] - 0s 489us/sample - loss: 0.6470 -
accuracy: 0.7378
Epoch 51/100
740/740 [==============================] - 0s 399us/sample - loss: 0.6224 -
accuracy: 0.7405
Epoch 52/100
740/740 [==============================] - 0s 404us/sample - loss: 0.6340 -
accuracy: 0.7514
Epoch 53/100
740/740 [==============================] - 0s 394us/sample - loss: 0.6089 -
accuracy: 0.7432
Epoch 54/100
740/740 [==============================] - 0s 386us/sample - loss: 0.6063 -
accuracy: 0.7568
Epoch 55/100
740/740 [==============================] - 0s 421us/sample - loss: 0.5929 -
accuracy: 0.7541
Epoch 56/100
740/740 [==============================] - 0s 375us/sample - loss: 0.5928 -
accuracy: 0.7554
Epoch 57/100
740/740 [==============================] - ETA: 0s - loss: 0.5939 - accuracy:
0.74 - 0s 397us/sample - loss: 0.5882 - accuracy: 0.7405
Epoch 58/100
740/740 [==============================] - 0s 415us/sample - loss: 0.5965 -
accuracy: 0.7662
Epoch 59/100
740/740 [==============================] - 0s 425us/sample - loss: 0.6308 -
accuracy: 0.7473
Epoch 60/100
740/740 [==============================] - 0s 365us/sample - loss: 0.5771 -
accuracy: 0.7662
Epoch 61/100
740/740 [==============================] - 0s 388us/sample - loss: 0.5840 -
```

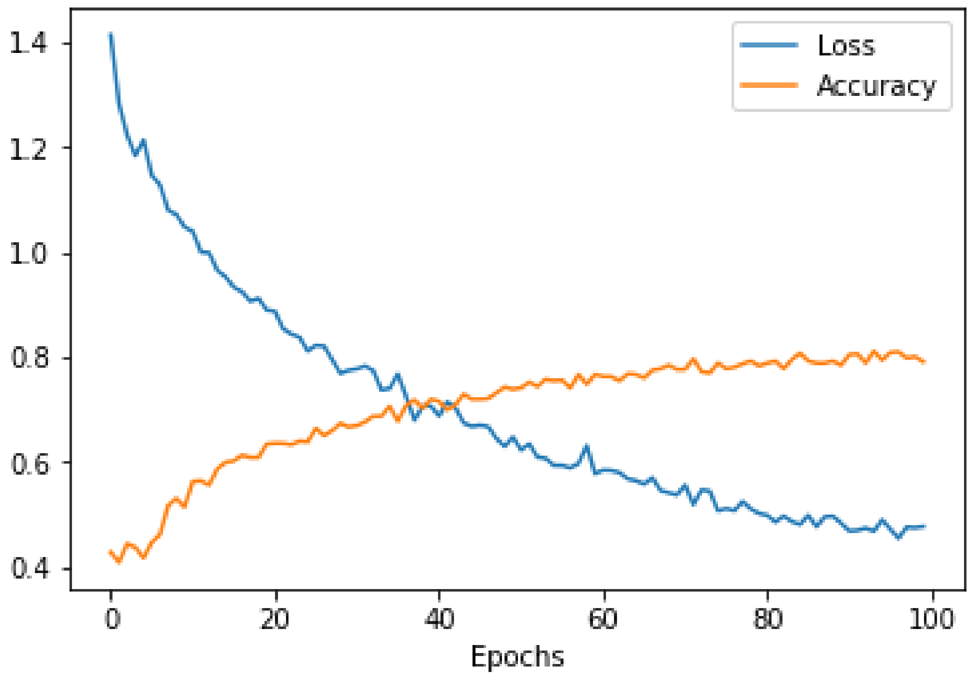
```
accuracy: 0.7622
Epoch 62/100
740/740 [==============================] - 0s 375us/sample - loss: 0.5833 -
accuracy: 0.7622
Epoch 63/100
740/740 [==============================] - 0s 412us/sample - loss: 0.5791 -
accuracy: 0.7541
Epoch 64/100
740/740 [==============================] - 0s 380us/sample - loss: 0.5671 -
accuracy: 0.7676
Epoch 65/100
740/740 [==============================] - 0s 403us/sample - loss: 0.5638 -
accuracy: 0.7662
Epoch 66/100
740/740 [==============================] - 0s 365us/sample - loss: 0.5570 -
accuracy: 0.7595
Epoch 67/100
740/740 [==============================] - 0s 377us/sample - loss: 0.5694 -
accuracy: 0.7743
Epoch 68/100
740/740 [==============================] - 0s 393us/sample - loss: 0.5448 -
accuracy: 0.7770
Epoch 69/100
740/740 [==============================] - 0s 389us/sample - loss: 0.5406 -
accuracy: 0.7838
Epoch 70/100
740/740 [==============================] - 0s 386us/sample - loss: 0.5365 -
accuracy: 0.7757
Epoch 71/100
740/740 [==============================] - 0s 384us/sample - loss: 0.5554 -
accuracy: 0.7757
Epoch 72/100
740/740 [==============================] - 0s 375us/sample - loss: 0.5181 -
accuracy: 0.7959
Epoch 73/100
740/740 [==============================] - 0s 374us/sample - loss: 0.5460 -
accuracy: 0.7716
Epoch 74/100
740/740 [==============================] - 0s 381us/sample - loss: 0.5440 -
accuracy: 0.7689
Epoch 75/100
740/740 [==============================] - 0s 381us/sample - loss: 0.5063 -
accuracy: 0.7878
Epoch 76/100
740/740 [==============================] - 0s 425us/sample - loss: 0.5105 -
accuracy: 0.7770
Epoch 77/100
740/740 [==============================] - 0s 427us/sample - loss: 0.5065 -
```

```
accuracy: 0.7797
Epoch 78/100
740/740 [==============================] - 0s 416us/sample - loss: 0.5239 -
accuracy: 0.7865
Epoch 79/100
740/740 [==============================] - 0s 393us/sample - loss: 0.5096 -
accuracy: 0.7919
Epoch 80/100
740/740 [==============================] - 0s 415us/sample - loss: 0.5012 -
accuracy: 0.7824
Epoch 81/100
740/740 [==============================] - 0s 384us/sample - loss: 0.4976 -
accuracy: 0.7878
Epoch 82/100
740/740 [==============================] - 0s 424us/sample - loss: 0.4848 -
accuracy: 0.7919
Epoch 83/100
740/740 [==============================] - 0s 414us/sample - loss: 0.4963 -
accuracy: 0.7770
Epoch 84/100
740/740 [==============================] - 0s 422us/sample - loss: 0.4858 -
accuracy: 0.7946
Epoch 85/100
740/740 [==============================] - 0s 418us/sample - loss: 0.4800 -
accuracy: 0.8068
Epoch 86/100
740/740 [==============================] - 0s 429us/sample - loss: 0.4978 -
accuracy: 0.7919
Epoch 87/100
740/740 [==============================] - 0s 418us/sample - loss: 0.4756 -
accuracy: 0.7878
Epoch 88/100
740/740 [==============================] - 0s 420us/sample - loss: 0.4948 -
accuracy: 0.7878
Epoch 89/100
740/740 [==============================] - 0s 419us/sample - loss: 0.4961 -
accuracy: 0.7919
Epoch 90/100
740/740 [==============================] - 0s 420us/sample - loss: 0.4827 -
accuracy: 0.7838
Epoch 91/100
740/740 [==============================] - 0s 427us/sample - loss: 0.4678 -
accuracy: 0.8041
Epoch 92/100
740/740 [==============================] - 0s 414us/sample - loss: 0.4688 -
accuracy: 0.8054
Epoch 93/100
740/740 [==============================] - 0s 424us/sample - loss: 0.4734 -
```

```
accuracy: 0.7878
Epoch 94/100
740/740 [==============================] - 0s 409us/sample - loss: 0.4671 -
accuracy: 0.8108
Epoch 95/100
740/740 [==============================] - 0s 429us/sample - loss: 0.4889 -
accuracy: 0.7932
Epoch 96/100
740/740 [==============================] - 0s 422us/sample - loss: 0.4713 -
accuracy: 0.8081
Epoch 97/100
740/740 [==============================] - 0s 414us/sample - loss: 0.4537 -
accuracy: 0.8095
Epoch 98/100
740/740 [==============================] - 0s 410us/sample - loss: 0.4753 -
accuracy: 0.7973
Epoch 99/100
740/740 [==============================] - 0s 408us/sample - loss: 0.4740 -
accuracy: 0.8000
Epoch 100/100
740/740 [==============================] - 0s 422us/sample - loss: 0.4762 -
accuracy: 0.7905
```

[88]:

```
pyplot.plot(history_Emb5.history['loss'])
pyplot.plot(history_Emb5.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



[89]:
```
val_loss_test10, val_acc_test10 = model_Emb5.evaluate([X5_test_Cat,␣
 ↪X5_test_Num], Y5_test)
val_acc_test.append(val_acc_test10)

print('Test accuracy:', val_acc_test10)
print('Test loss:', val_loss_test10)

val_loss_train10, val_acc_train10 = model_Emb5.evaluate([X5_train_Cat,␣
 ↪X5_train_Num], Y5_train)

print('Train accuracy:', val_acc_train10)
print('Train loss:', val_loss_train10)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
365/365 [==============================] - 0s 1ms/sample - loss: 1.9901 -
accuracy: 0.5425
Test accuracy: 0.54246575
Test loss: 1.99007576181464
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 193us/sample - loss: 0.3712 -
accuracy: 0.8486
```

```
Train accuracy: 0.84864867
Train loss: 0.3712417922309927
```

Combining general quoted article feature representations with other features in the GDPR dataframe (GA+OF)

[90]:
```
df_OF_GA_Df = pd.concat([df_Cat_Gen_GDPR_Article_Df, df_OF], axis=1).
  ↪reindex(df_Cat_GDPR_Article_Df.index)
df_OF_GA_Df = df_OF_GA_Df.drop(['Id'], axis = 1)
```

[91]:
```
X6 = df_OF_GA_Df.iloc[:,0:len(df_OF_GA_Df.columns)-1]
Y6 = df_OF_GA_Df['Fine_binned1']
X6_train, X6_test, Y6_train, Y6_test = train_test_split(X6, Y6, test_size=0.33,␣
  ↪random_state=42)

X6_train_Cat= X6_train.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]
X6_test_Cat= X6_test.iloc[:,0:len(df_Cat_Gen_GDPR_Article_Df.columns)-1]

X6_train_Num= X6_train.iloc[:,len(df_Cat_Gen_GDPR_Article_Df.columns)-1:len(X6.
  ↪columns)]
X6_test_Num= X6_test.iloc[:,len(df_Cat_Gen_GDPR_Article_Df.columns)-1:len(X6.
  ↪columns)]
```

[92]:
```
# Use Input layers, specify input shape
no_of_unique_cat_gen= len(df_Cat_Gen_GDPR_Article_Df.columns)-1
inp_num_data = keras.layers.Input(shape=(5,))

# Creating the model
model_Emb6 = model_emb_def(no_of_unique_cat_gen,inp_num_data)
model_Emb6.summary()
```

```
Model: "model_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_4 (InputLayer)            [(None, 62)]         0
________________________________________________________________________________
__________________
embedding_1 (Embedding)         (None, 62, 31)       1922        input_4[0][0]
________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 1922)         0
embedding_1[0][0]
________________________________________________________________________________
__________________
input_3 (InputLayer)            [(None, 5)]          0
```

```
--------------------------------------------------------------------------------
------------------
concatenate_1 (Concatenate)     (None, 1927)         0           flatten_1[0][0]
                                                                 input_3[0][0]

--------------------------------------------------------------------------------
------------------
dense_5 (Dense)                 (None, 50)           96400
concatenate_1[0][0]

--------------------------------------------------------------------------------
------------------
dense_6 (Dense)                 (None, 25)           1275        dense_5[0][0]

--------------------------------------------------------------------------------
------------------
dropout_2 (Dropout)             (None, 25)           0           dense_6[0][0]

--------------------------------------------------------------------------------
------------------
dense_7 (Dense)                 (None, 15)           390         dropout_2[0][0]

--------------------------------------------------------------------------------
------------------
dropout_3 (Dropout)             (None, 15)           0           dense_7[0][0]

--------------------------------------------------------------------------------
------------------
dense_8 (Dense)                 (None, 10)           160         dropout_3[0][0]

--------------------------------------------------------------------------------
------------------
dense_9 (Dense)                 (None, 5)            55          dense_8[0][0]
================================================================================
==================
Total params: 100,202
Trainable params: 100,202
Non-trainable params: 0

--------------------------------------------------------------------------------
------------------
```

[93]:

```
model_Emb6.compile(loss='sparse_categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])
history_Emb6 = model_Emb6.fit([X6_train_Cat, X6_train_Num],␣
 ↪Y6_train,epochs=100, batch_size=32)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
Train on 740 samples
Epoch 1/100
740/740 [==============================] - 1s 1ms/sample - loss: 1.3582 -
accuracy: 0.3568
Epoch 2/100
740/740 [==============================] - 0s 229us/sample - loss: 1.1670 -
```

```
accuracy: 0.4608
Epoch 3/100
740/740 [==============================] - 0s 225us/sample - loss: 1.1583 -
accuracy: 0.4635
Epoch 4/100
740/740 [==============================] - 0s 234us/sample - loss: 1.1577 -
accuracy: 0.4284
Epoch 5/100
740/740 [==============================] - 0s 248us/sample - loss: 1.1396 -
accuracy: 0.4122
Epoch 6/100
740/740 [==============================] - 0s 221us/sample - loss: 1.1374 -
accuracy: 0.4514
Epoch 7/100
740/740 [==============================] - 0s 219us/sample - loss: 1.1138 -
accuracy: 0.4608
Epoch 8/100
740/740 [==============================] - 0s 224us/sample - loss: 1.1036 -
accuracy: 0.4554
Epoch 9/100
740/740 [==============================] - 0s 235us/sample - loss: 1.1098 -
accuracy: 0.4473
Epoch 10/100
740/740 [==============================] - 0s 224us/sample - loss: 1.0952 -
accuracy: 0.4311 - loss: 1.0882 - accuracy: 0.43
Epoch 11/100
740/740 [==============================] - 0s 224us/sample - loss: 1.0758 -
accuracy: 0.4730
Epoch 12/100
740/740 [==============================] - 0s 221us/sample - loss: 1.0765 -
accuracy: 0.4459
Epoch 13/100
740/740 [==============================] - 0s 233us/sample - loss: 1.0878 -
accuracy: 0.4811
Epoch 14/100
740/740 [==============================] - 0s 221us/sample - loss: 1.0703 -
accuracy: 0.4662
Epoch 15/100
740/740 [==============================] - 0s 228us/sample - loss: 1.0519 -
accuracy: 0.4757
Epoch 16/100
740/740 [==============================] - 0s 225us/sample - loss: 1.0733 -
accuracy: 0.4784
Epoch 17/100
740/740 [==============================] - 0s 210us/sample - loss: 1.0685 -
accuracy: 0.4878
Epoch 18/100
740/740 [==============================] - 0s 222us/sample - loss: 1.0427 -
```

```
accuracy: 0.4527
Epoch 19/100
740/740 [==============================] - 0s 209us/sample - loss: 1.0299 -
accuracy: 0.4851
Epoch 20/100
740/740 [==============================] - 0s 221us/sample - loss: 1.0187 -
accuracy: 0.4959
Epoch 21/100
740/740 [==============================] - 0s 221us/sample - loss: 1.0155 -
accuracy: 0.4905
Epoch 22/100
740/740 [==============================] - 0s 228us/sample - loss: 1.0005 -
accuracy: 0.5041
Epoch 23/100
740/740 [==============================] - 0s 219us/sample - loss: 0.9879 -
accuracy: 0.5257
Epoch 24/100
740/740 [==============================] - 0s 244us/sample - loss: 0.9815 -
accuracy: 0.5189
Epoch 25/100
740/740 [==============================] - 0s 216us/sample - loss: 0.9690 -
accuracy: 0.5108
Epoch 26/100
740/740 [==============================] - 0s 214us/sample - loss: 0.9641 -
accuracy: 0.5365
Epoch 27/100
740/740 [==============================] - 0s 220us/sample - loss: 0.9533 -
accuracy: 0.5365
Epoch 28/100
740/740 [==============================] - 0s 216us/sample - loss: 0.9394 -
accuracy: 0.5432
Epoch 29/100
740/740 [==============================] - 0s 225us/sample - loss: 0.9686 -
accuracy: 0.5554
Epoch 30/100
740/740 [==============================] - 0s 218us/sample - loss: 0.9344 -
accuracy: 0.5473
Epoch 31/100
740/740 [==============================] - 0s 210us/sample - loss: 0.9285 -
accuracy: 0.5608
Epoch 32/100
740/740 [==============================] - 0s 216us/sample - loss: 0.9290 -
accuracy: 0.5459
Epoch 33/100
740/740 [==============================] - 0s 214us/sample - loss: 0.9433 -
accuracy: 0.5770
Epoch 34/100
740/740 [==============================] - 0s 221us/sample - loss: 0.9198 -
```

```
accuracy: 0.5676
Epoch 35/100
740/740 [==============================] - 0s 216us/sample - loss: 0.8929 -
accuracy: 0.5703
Epoch 36/100
740/740 [==============================] - 0s 217us/sample - loss: 0.8826 -
accuracy: 0.5851
Epoch 37/100
740/740 [==============================] - 0s 214us/sample - loss: 0.8654 -
accuracy: 0.5865
Epoch 38/100
740/740 [==============================] - 0s 194us/sample - loss: 0.8723 -
accuracy: 0.5905
Epoch 39/100
740/740 [==============================] - 0s 237us/sample - loss: 0.8632 -
accuracy: 0.5865
Epoch 40/100
740/740 [==============================] - 0s 208us/sample - loss: 0.8508 -
accuracy: 0.6054
Epoch 41/100
740/740 [==============================] - 0s 209us/sample - loss: 0.8245 -
accuracy: 0.6149
Epoch 42/100
740/740 [==============================] - 0s 210us/sample - loss: 0.8124 -
accuracy: 0.6054
Epoch 43/100
740/740 [==============================] - 0s 208us/sample - loss: 0.8118 -
accuracy: 0.6216
Epoch 44/100
740/740 [==============================] - 0s 268us/sample - loss: 0.7823 -
accuracy: 0.6270
Epoch 45/100
740/740 [==============================] - 0s 224us/sample - loss: 0.7661 -
accuracy: 0.6662
Epoch 46/100
740/740 [==============================] - 0s 208us/sample - loss: 0.7828 -
accuracy: 0.6392
Epoch 47/100
740/740 [==============================] - 0s 241us/sample - loss: 0.7634 -
accuracy: 0.6500
Epoch 48/100
740/740 [==============================] - 0s 216us/sample - loss: 0.7810 -
accuracy: 0.6311 - loss: 0.7825 - accuracy: 0.
Epoch 49/100
740/740 [==============================] - 0s 201us/sample - loss: 0.7618 -
accuracy: 0.6568
Epoch 50/100
740/740 [==============================] - 0s 207us/sample - loss: 0.7523 -
```

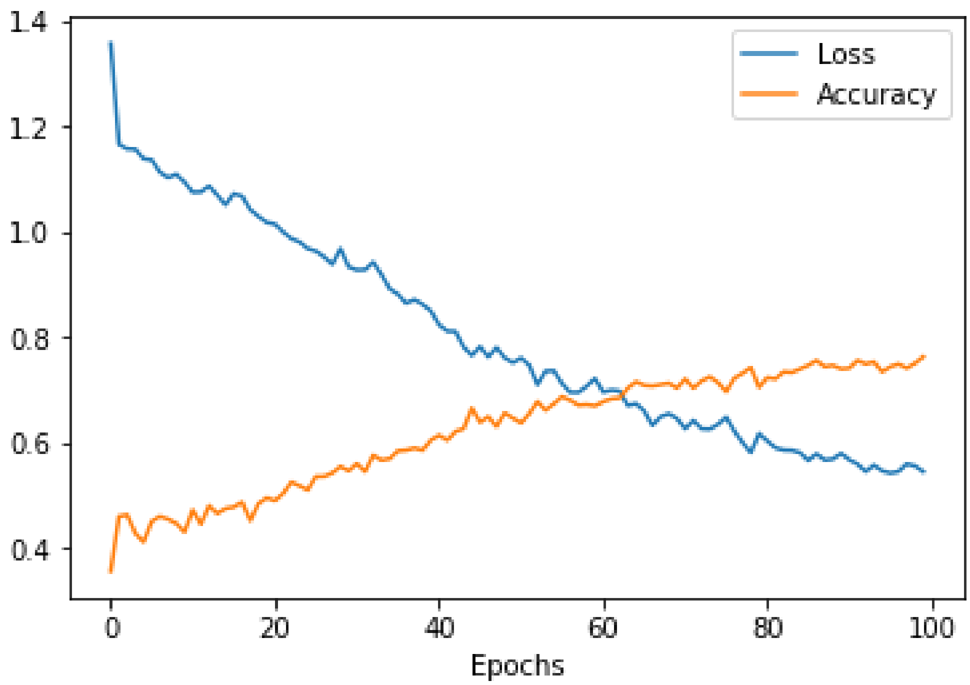
```
accuracy: 0.6473
Epoch 51/100
740/740 [==============================] - 0s 206us/sample - loss: 0.7614 -
accuracy: 0.6378
Epoch 52/100
740/740 [==============================] - 0s 200us/sample - loss: 0.7479 -
accuracy: 0.6554
Epoch 53/100
740/740 [==============================] - 0s 210us/sample - loss: 0.7107 -
accuracy: 0.6784
Epoch 54/100
740/740 [==============================] - 0s 269us/sample - loss: 0.7364 -
accuracy: 0.6622
Epoch 55/100
740/740 [==============================] - 0s 210us/sample - loss: 0.7372 -
accuracy: 0.6743
Epoch 56/100
740/740 [==============================] - 0s 210us/sample - loss: 0.7128 -
accuracy: 0.6878
Epoch 57/100
740/740 [==============================] - 0s 202us/sample - loss: 0.6961 -
accuracy: 0.6811
Epoch 58/100
740/740 [==============================] - 0s 219us/sample - loss: 0.6960 -
accuracy: 0.6716
Epoch 59/100
740/740 [==============================] - 0s 212us/sample - loss: 0.7078 -
accuracy: 0.6730
Epoch 60/100
740/740 [==============================] - 0s 208us/sample - loss: 0.7225 -
accuracy: 0.6703
Epoch 61/100
740/740 [==============================] - 0s 198us/sample - loss: 0.6965 -
accuracy: 0.6784
Epoch 62/100
740/740 [==============================] - 0s 199us/sample - loss: 0.7004 -
accuracy: 0.6838
Epoch 63/100
740/740 [==============================] - 0s 193us/sample - loss: 0.6986 -
accuracy: 0.6851
Epoch 64/100
740/740 [==============================] - 0s 195us/sample - loss: 0.6709 -
accuracy: 0.7041
Epoch 65/100
740/740 [==============================] - 0s 199us/sample - loss: 0.6748 -
accuracy: 0.7162
Epoch 66/100
740/740 [==============================] - 0s 209us/sample - loss: 0.6608 -
```

```
accuracy: 0.7095
Epoch 67/100
740/740 [==============================] - 0s 202us/sample - loss: 0.6332 -
accuracy: 0.7081
Epoch 68/100
740/740 [==============================] - 0s 225us/sample - loss: 0.6497 -
accuracy: 0.7108
Epoch 69/100
740/740 [==============================] - 0s 185us/sample - loss: 0.6555 -
accuracy: 0.7135
Epoch 70/100
740/740 [==============================] - 0s 198us/sample - loss: 0.6465 -
accuracy: 0.7041
Epoch 71/100
740/740 [==============================] - 0s 195us/sample - loss: 0.6269 -
accuracy: 0.7216
Epoch 72/100
740/740 [==============================] - 0s 201us/sample - loss: 0.6427 -
accuracy: 0.7041
Epoch 73/100
740/740 [==============================] - 0s 191us/sample - loss: 0.6269 -
accuracy: 0.7176
Epoch 74/100
740/740 [==============================] - 0s 191us/sample - loss: 0.6261 -
accuracy: 0.7257
Epoch 75/100
740/740 [==============================] - 0s 183us/sample - loss: 0.6352 -
accuracy: 0.7149
Epoch 76/100
740/740 [==============================] - 0s 209us/sample - loss: 0.6489 -
accuracy: 0.6973
Epoch 77/100
740/740 [==============================] - 0s 209us/sample - loss: 0.6225 -
accuracy: 0.7230
Epoch 78/100
740/740 [==============================] - 0s 220us/sample - loss: 0.6006 -
accuracy: 0.7324
Epoch 79/100
740/740 [==============================] - 0s 209us/sample - loss: 0.5819 -
accuracy: 0.7432
Epoch 80/100
740/740 [==============================] - 0s 220us/sample - loss: 0.6180 -
accuracy: 0.7054
Epoch 81/100
740/740 [==============================] - 0s 224us/sample - loss: 0.6040 -
accuracy: 0.7243
Epoch 82/100
740/740 [==============================] - 0s 209us/sample - loss: 0.5909 -
```

```
accuracy: 0.7216
Epoch 83/100
740/740 [==============================] - 0s 230us/sample - loss: 0.5867 -
accuracy: 0.7351
Epoch 84/100
740/740 [==============================] - 0s 214us/sample - loss: 0.5863 -
accuracy: 0.7338
Epoch 85/100
740/740 [==============================] - 0s 224us/sample - loss: 0.5817 -
accuracy: 0.7405
Epoch 86/100
740/740 [==============================] - 0s 214us/sample - loss: 0.5669 -
accuracy: 0.7473
Epoch 87/100
740/740 [==============================] - 0s 208us/sample - loss: 0.5797 -
accuracy: 0.7568
Epoch 88/100
740/740 [==============================] - 0s 209us/sample - loss: 0.5678 -
accuracy: 0.7446
Epoch 89/100
740/740 [==============================] - 0s 232us/sample - loss: 0.5696 -
accuracy: 0.7473
Epoch 90/100
740/740 [==============================] - 0s 204us/sample - loss: 0.5805 -
accuracy: 0.7405
Epoch 91/100
740/740 [==============================] - 0s 214us/sample - loss: 0.5683 -
accuracy: 0.7419
Epoch 92/100
740/740 [==============================] - 0s 214us/sample - loss: 0.5601 -
accuracy: 0.7568
Epoch 93/100
740/740 [==============================] - 0s 212us/sample - loss: 0.5458 -
accuracy: 0.7500
Epoch 94/100
740/740 [==============================] - 0s 213us/sample - loss: 0.5581 -
accuracy: 0.7541
Epoch 95/100
740/740 [==============================] - 0s 209us/sample - loss: 0.5473 -
accuracy: 0.7351
Epoch 96/100
740/740 [==============================] - 0s 213us/sample - loss: 0.5431 -
accuracy: 0.7446
Epoch 97/100
740/740 [==============================] - 0s 218us/sample - loss: 0.5459 -
accuracy: 0.7500
Epoch 98/100
740/740 [==============================] - 0s 213us/sample - loss: 0.5604 -
```

```
accuracy: 0.7419
Epoch 99/100
740/740 [==============================] - 0s 218us/sample - loss: 0.5565 -
accuracy: 0.7514
Epoch 100/100
740/740 [==============================] - 0s 209us/sample - loss: 0.5459 -
accuracy: 0.7635
```

[94]:
```
pyplot.plot(history_Emb6.history['loss'])
pyplot.plot(history_Emb6.history['accuracy'])
pyplot.xlabel("Epochs")
pyplot.legend(['Loss', 'Accuracy'])
pyplot.show()
```



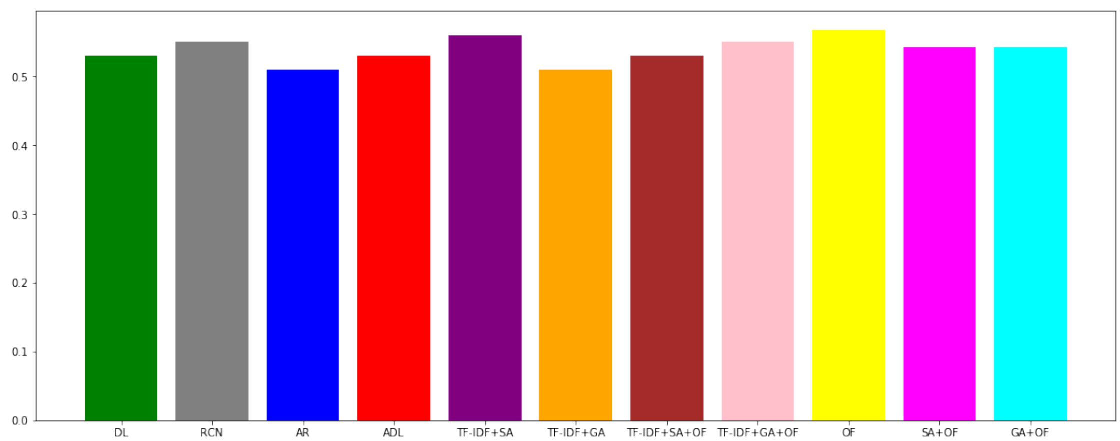
[95]:
```
val_loss_test11, val_acc_test11 = model_Emb6.evaluate([X6_test_Cat,␣
 ↪X6_test_Num], Y6_test)
val_acc_test.append(val_acc_test11)

print('Test accuracy:', val_acc_test11)
print('Test loss:', val_loss_test11)

val_loss_train11, val_acc_train11 = model_Emb6.evaluate([X6_train_Cat,␣
 ↪X6_train_Num], Y6_train)
```

```
print('Train accuracy:', val_acc_train11)
print('Train loss:', val_loss_train11)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
365/365 [==============================] - 0s 945us/sample - loss: 1.6107 -
accuracy: 0.5425
Test accuracy: 0.54246575
Test loss: 1.6106953426583173
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find
data adapter that can handle input: (<class 'list'> containing values of types
{"<class 'pandas.core.frame.DataFrame'>"}), <class 'NoneType'>
740/740 [==============================] - 0s 115us/sample - loss: 0.4592 -
accuracy: 0.8027
Train accuracy: 0.8027027
Train loss: 0.4591899951164787
```

[98]:
```
colors2 = ['green', 'grey', 'blue',␣
 ↪'red','purple','orange','brown','pink','yellow','magenta','cyan']
names2 =  ['DL', 'RCN', 'AR',␣
 ↪'ADL','TF-IDF+SA','TF-IDF+GA','TF-IDF+SA+OF','TF-IDF+GA+OF','OF',␣
 ↪'SA+OF','GA+OF']
pyplot.figure(figsize=(18, 7))
pyplot.bar(names2, val_acc_test,color=colors2)
```

[98]: <BarContainer object of 11 artists>



8 Conclusions

- We have looked at some approaches for predicting the risk of a company getting fined.

- The experimental results show that the deep learning model that combines TF-IDF and specific quoted article feature representations obtains the best performance.
- However, deeper feature extraction like a tree representation of the quoted article feature could help improve the model performance.