Bank Authentication Data Set

- from the UCI Repository

NN Project

In [22]:
```
import pandas as pd
import seaborn as sns
%matplotlib inline
```

In [23]:
```
data = pd.read_csv('bank_note_data.csv')
```

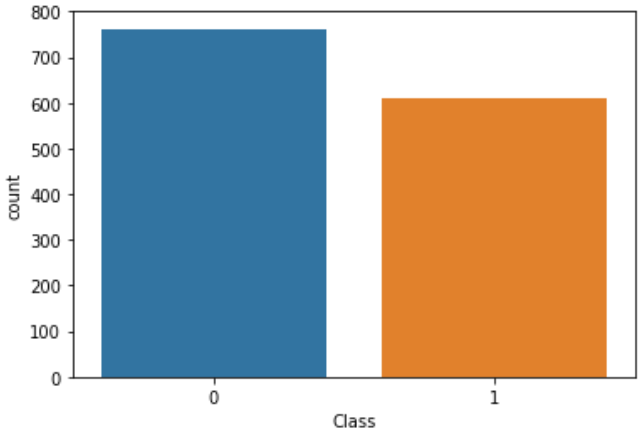
In [24]:
```
data.head()
```

Out[24]:

	Image.Var	Image.Skew	Image.Curt	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

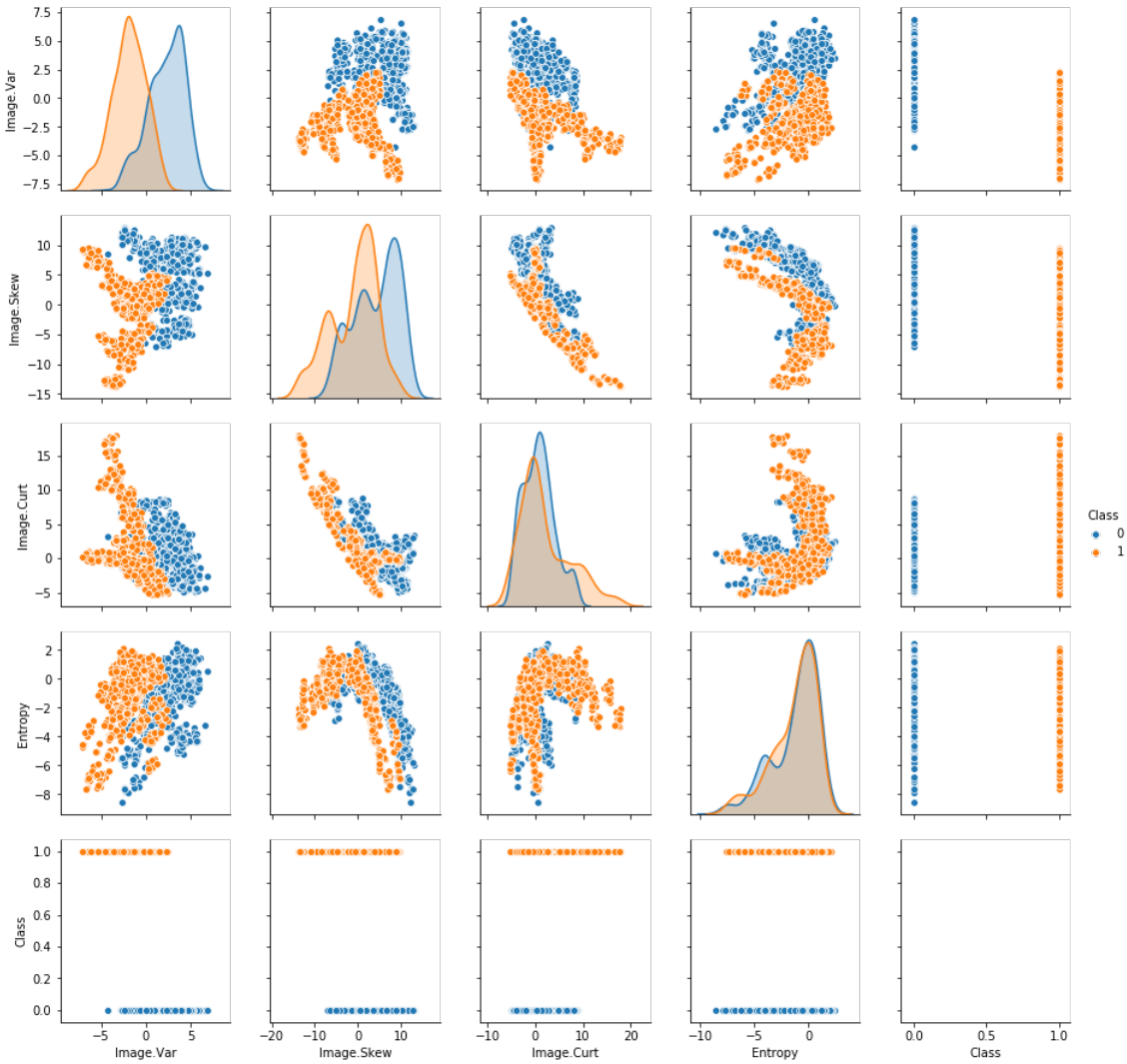
In [25]:
```
sns.countplot(x='Class', data=data)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3fb8cad0>
```



In [27]:
```
#Take a look at the data
sns.pairplot(data,hue='Class')
```

Out[27]: <seaborn.axisgrid.PairGrid at 0x1a407f5b90>



In [29]:
```
# Standard Scaling
from sklearn.preprocessing import StandardScaler
```

In [30]:
```
# Create an object called scaler
scaler = StandardScaler()
```

In [31]:
```
# Fit scaler to features
scaler.fit(data.drop('Class', axis=1))
```

Out[31]: StandardScaler(copy=True, with_mean=True, with_std=True)

In [32]:
```
# Use transform method to transform the features to a scaled version
scaled_feature = scaler.transform(data.drop('Class', axis=1))
```

In [33]:
```
# Convert scaled features to a dataframe and check the head to make sure it worke
d
df_feat = pd.DataFrame(scaled_feature,columns = data.columns[:-1])
df_feat.head()
```

Out[33]:

	Image.Var	Image.Skew	Image.Curt	Entropy
0	1.121806	1.149455	-0.975970	0.354561
1	1.447066	1.064453	-0.895036	-0.128767
2	1.207810	-0.777352	0.122218	0.618073
3	1.063742	1.295478	-1.255397	-1.144029
4	-0.036772	-1.087038	0.736730	0.096587

In [34]:
```
# Train Test Split

X = df_feat

y = data['Class']
```

In [35]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [36]:
```
# TensorFlow
import tensorflow as tf
```

In [37]:
```
# Create a list of feature Columns
df_feat.columns
```

Out[37]:
```
Index(['Image.Var', 'Image.Skew', 'Image.Curt', 'Entropy'], dtype='object')
```

In [38]:
```
feat_cols = []

for col in df_feat.columns:
    feat_cols.append(tf.feature_column.numeric_column(col))
```

In [39]:
```
feat_cols
```

Out[39]:
```
[NumericColumn(key='Image.Var', shape=(1,), default_value=None, dtype=tf.float3
2, normalizer_fn=None),
 NumericColumn(key='Image.Skew', shape=(1,), default_value=None, dtype=tf.float3
2, normalizer_fn=None),
 NumericColumn(key='Image.Curt', shape=(1,), default_value=None, dtype=tf.float3
2, normalizer_fn=None),
 NumericColumn(key='Entropy', shape=(1,), default_value=None, dtype=tf.float32,
normalizer_fn=None)]
```

In [40]:
```
# Create an object called classifier. Set 2 classes [10,20,10] hidden unit layer
structure
classifier = tf.estimator.DNNClassifier(hidden_units = [10,20,10],
                                        n_classes = 2,
                                        feature_columns=feat_cols)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0828 14:19:50.097635 140736573572032 estimator.py:1811] Using temporary folder
as model directory: /var/folders/bg/2b17ybm53nz575_6xtq1zxnh0000gn/T/tmp2bwoojez
```

In [41]:
```
# Create a tf.estimator.pandas_input train and batch size

input_func = tf.estimator.inputs.pandas_input_fn(x = X_train,
                                                 y = y_train,
                                                 batch_size = 20,
                                                shuffle = True)
```

In [43]:
```
# Train classifer to the input function

classifier.train(input_fn = input_func, steps=500)
```

```
W0828 14:28:02.788529 140736573572032 deprecation.py:323] From /anaconda3/lib/py
thon3.7/site-packages/tensorflow/python/training/saver.py:1276: checkpoint_exist
s (from tensorflow.python.training.checkpoint_management) is deprecated and will
be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
W0828 14:28:02.892272 140736573572032 deprecation.py:323] From /anaconda3/lib/py
thon3.7/site-packages/tensorflow/python/training/saver.py:1066: get_checkpoint_m
times (from tensorflow.python.training.checkpoint_management) is deprecated and
will be removed in a future version.
Instructions for updating:
Use standard file utilities to get mtimes.
```

Out[43]:
```
<tensorflow_estimator.python.estimator.canned.dnn.DNNClassifier at 0x1a4167d510>
```

Model Evaluation

In [44]:
```
pred_fn = tf.estimator.inputs.pandas_input_fn(x=X_test,
                                              batch_size = len(X_test),
                                             shuffle = False)
```

In [45]:
```
note_predictions = list(classifier.predict(input_fn = pred_fn))
```

In [46]:
```
note_predictions[0]
```

Out[46]:
```
{'logits': array([-13.71891], dtype=float32),
 'logistic': array([1.1026859e-06], dtype=float32),
 'probabilities': array([9.999989e-01, 1.101419e-06], dtype=float32),
 'class_ids': array([0]),
 'classes': array([b'0'], dtype=object),
 'all_class_ids': array([0, 1], dtype=int32),
 'all_classes': array([b'0', b'1'], dtype=object)}
```

In [47]:
```
final_preds  = []
for pred in note_predictions:
    final_preds.append(pred['class_ids'][0])
```

In [48]:
```
#Build the reports
from sklearn.metrics import classification_report,confusion_matrix
```

In [49]:
```
print(confusion_matrix(y_test,final_preds))
```

```
[[231   0]
 [  0 181]]
```

In [50]:
```
print(classification_report(y_test,final_preds))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       231
           1       1.00      1.00      1.00       181

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412
```

Create a Random Forest Classifier and compare the confusion matrix and classification report to the DNN model

In [51]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [55]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

In [56]:
```
rfc.fit(X_train,y_train)
```

Out[56]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [57]:
```
rfc_preds = rfc.predict(X_test)
```

In [58]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

In [59]:
```
print(classification_report(y_test,rfc_preds))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       231
           1       0.98      0.99      0.99       181

    accuracy                           0.99       412
   macro avg       0.99      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

In [60]:
```
print(confusion_matrix(y_test,rfc_preds))
```

```
[[228   3]
 [  1 180]]
```

In [ ]:

In [ ]: