

```
[24]: import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, Normalizer
from sklearn.decomposition import PCA as sklearnPCA

import warnings
warnings.filterwarnings("ignore")
```

```
[25]: wbcd = pd.read_csv("../input/data.csv")
wbcd.head()
```

```
[25]:
```

	id	diagnosis	...	fractal_dimension_worst	Unnamed: 32
0	842302	M	...	0.11890	NaN
1	842517	M	...	0.08902	NaN
2	84300903	M	...	0.08758	NaN
3	84348301	M	...	0.17300	NaN
4	84358402	M	...	0.07678	NaN

[5 rows x 33 columns]

```
[26]: print("This WBCD dataset is consisted of",wbcd.shape)
```

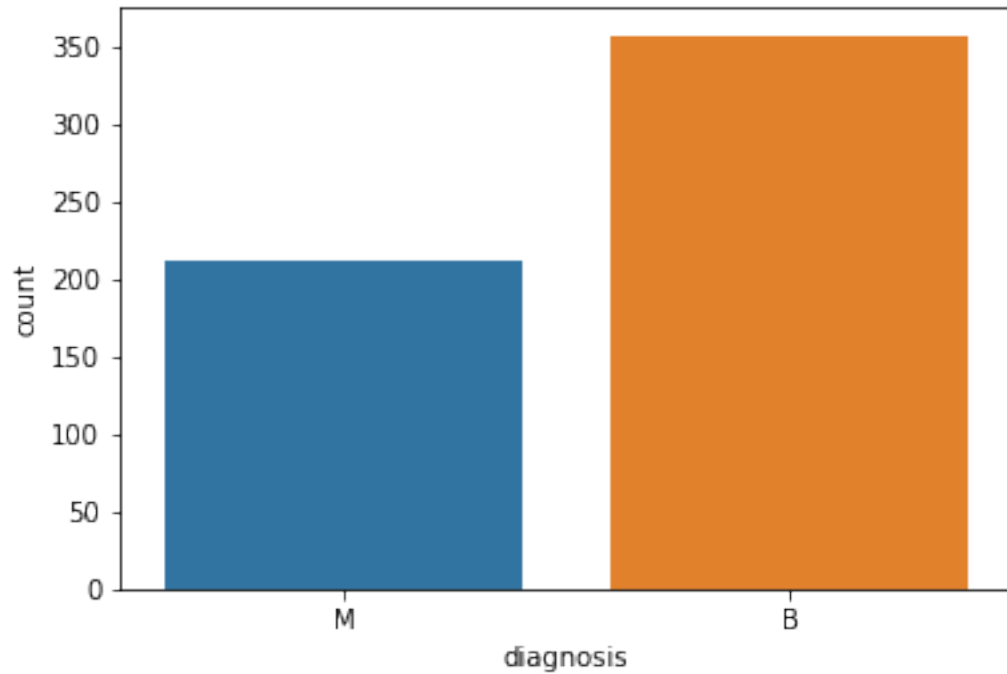
This WBCD dataset is consisted of (569, 33)

```
[27]: wbcd = wbcd.iloc[:, :-1]
print("This WBCD dataset is consisted of",wbcd.shape)
```

This WBCD dataset is consisted of (569, 32)

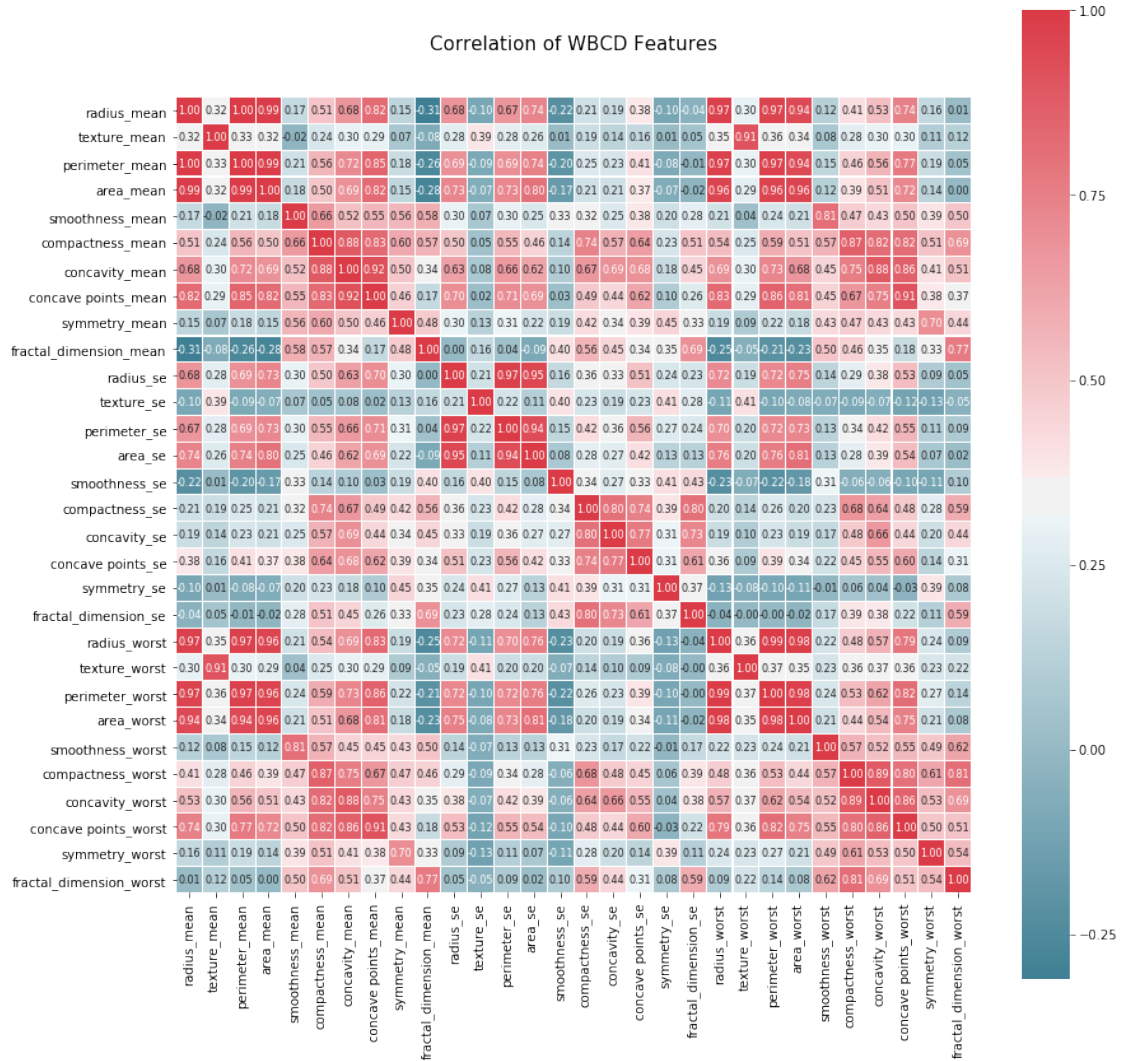
```
[28]: sns.countplot(wbcd['diagnosis'],label="Count")
```

```
[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7ef939838630>
```



```
[29]: corr = wbcd.iloc[:,2:].corr()
colormap = sns.diverging_palette(220, 10, as_cmap = True)
plt.figure(figsize=(14,14))
sns.heatmap(corr, cbar = True, square = True, annot=True, fmt= '.
↪2f',annot_kws={'size': 8},
            cmap = colormap, linewidths=0.1, linecolor='white')
plt.title('Correlation of WBCD Features', y=1.05, size=15)
```

```
[29]: Text(0.5,1.05,'Correlation of WBCD Features')
```



```
[30]: train,test = train_test_split(wbcd, test_size=0.3, random_state=42)
print("Training Data :",train.shape)
print("Testing Data :",test.shape)
```

Training Data : (398, 32)
Testing Data : (171, 32)

```
[31]: train_id = train['id']
test_id = test['id']

train_data = train.iloc[:,1:]
test_data = test.iloc[:,1:]

print("Training Data :",train_data.shape)
```

```
print("Testing Data :", test_data.shape)
```

Training Data : (398, 31)

Testing Data : (171, 31)

```
[32]: # Training Data
train_x = train_data.iloc[:,1:]
train_x = MinMaxScaler().fit_transform(train_x)
print("Training Data :", train_x.shape)

# Testing Data
test_x = test_data.iloc[:,1:]
test_x = MinMaxScaler().fit_transform(test_x)
print("Testing Data :", test_x.shape)
```

Training Data : (398, 30)

Testing Data : (171, 30)

```
[33]: # Training Data
train_y = train_data.iloc[:,1]
train_y[train_y=='M'] = 0
train_y[train_y=='B'] = 1
print("Training Data :", train_y.shape)

# Testing Data
test_y = test_data.iloc[:,1]
test_y[test_y=='M'] = 0
test_y[test_y=='B'] = 1
print("Testing Data :", test_y.shape)
```

Training Data : (398, 1)

Testing Data : (171, 1)

```
[34]: X = tf.placeholder(tf.float32, [None,30])
Y = tf.placeholder(tf.float32, [None, 1])
```

```
[35]: # weight
W = tf.Variable(tf.random_normal([30,1], seed=0), name='weight')

# bias
b = tf.Variable(tf.random_normal([1], seed=0), name='bias')
```

```
[36]: logits = tf.matmul(X,W) + b
```

```
[37]: hypothesis = tf.nn.sigmoid(logits)

cost_i = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,labels=Y)
```

```
cost = tf.reduce_mean(cost_i)
```

```
[38]: train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

```
[39]: prediction = tf.cast(hypothesis > 0.5, dtype=tf.float32)
correct_prediction = tf.equal(prediction, Y)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, dtype=tf.float32))
```

```
[40]: with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: train_x, Y: train_y})
        if step % 1000 == 0:
            loss, acc = sess.run([cost, accuracy], feed_dict={X: train_x, Y:
↳train_y})
            print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step, loss,
↳acc))

            train_acc = sess.run(accuracy, feed_dict={X: train_x, Y: train_y})
            test_acc, test_predict, test_correct = sess.
↳run([accuracy, prediction, correct_prediction], feed_dict={X: test_x, Y:
↳test_y})
            print("Model Prediction =", train_acc)
            print("Test Prediction =", test_acc)
```

```
Step:    0      Loss: 0.848      Acc: 39.70%
Step:  1000     Loss: 0.238      Acc: 91.21%
Step:  2000     Loss: 0.180      Acc: 94.72%
Step:  3000     Loss: 0.154      Acc: 96.23%
Step:  4000     Loss: 0.138      Acc: 96.98%
Step:  5000     Loss: 0.128      Acc: 97.49%
Step:  6000     Loss: 0.120      Acc: 97.74%
Step:  7000     Loss: 0.114      Acc: 97.99%
Step:  8000     Loss: 0.110      Acc: 98.24%
Step:  9000     Loss: 0.106      Acc: 98.24%
Step: 10000     Loss: 0.102      Acc: 98.24%
Model Prediction = 0.982412
Test Prediction = 0.947368
```

```
[41]: def ann_slp():
    print("====Data Summary====")
    print("Training Data :", train_x.shape)
    print("Testing Data :", test_x.shape)

    X = tf.placeholder(tf.float32, [None, 30])
```

```

Y = tf.placeholder(tf.float32, [None, 1])

W = tf.Variable(tf.random_normal([30,1], seed=0), name='weight')
b = tf.Variable(tf.random_normal([1], seed=0), name='bias')

logits = tf.matmul(X,W) + b
hypothesis = tf.nn.sigmoid(logits)

cost_i = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,labels=Y)
cost = tf.reduce_mean(cost_i)

train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

prediction = tf.cast(hypothesis > 0.5, dtype=tf.float32)
correct_prediction = tf.equal(prediction, Y)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, dtype=tf.float32))

print("\n=====Processing=====")
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: train_x, Y: train_y})
        if step % 1000 == 0:
            loss, acc = sess.run([cost, accuracy], feed_dict={X: train_x, Y:
↪ train_y})
            print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step,
↪ loss, acc))

            train_acc = sess.run(accuracy, feed_dict={X: train_x, Y: train_y})
            test_acc,test_predict,test_correct = sess.
↪run([accuracy,prediction,correct_prediction], feed_dict={X: test_x, Y:
↪test_y})

            print("\n=====Results=====")
            print("Model Prediction =", train_acc)
            print("Test Prediction =", test_acc)

            return train_acc,test_acc

ann_slp_train_acc, ann_slp_test_acc = ann_slp()

```

=====Data Summary=====

Training Data : (398, 30)

Testing Data : (171, 30)

=====Processing=====

Step: 0 Loss: 0.848 Acc: 39.70%

Step: 1000	Loss: 0.238	Acc: 91.21%
Step: 2000	Loss: 0.180	Acc: 94.72%
Step: 3000	Loss: 0.154	Acc: 96.23%
Step: 4000	Loss: 0.138	Acc: 96.98%
Step: 5000	Loss: 0.128	Acc: 97.49%
Step: 6000	Loss: 0.120	Acc: 97.74%
Step: 7000	Loss: 0.114	Acc: 97.99%
Step: 8000	Loss: 0.110	Acc: 98.24%
Step: 9000	Loss: 0.106	Acc: 98.24%
Step: 10000	Loss: 0.102	Acc: 98.24%

=====Results=====

Model Prediction = 0.982412

Test Prediction = 0.947368

```
[42]: def ann_slp_pca():
    sklearn_pca = sklearnPCA(n_components=10)

    print("=====Data Summary=====")
    pca_train_x = sklearn_pca.fit_transform(train_x)
    print("PCA Training Data :", pca_train_x.shape)

    pca_test_x = sklearn_pca.fit_transform(test_x)
    print("PCA Testing Data :", pca_test_x.shape)

    X = tf.placeholder(tf.float32, [None,10])
    Y = tf.placeholder(tf.float32, [None, 1])

    W = tf.Variable(tf.random_normal([10,1], seed=0), name='weight')
    b = tf.Variable(tf.random_normal([1], seed=0), name='bias')

    logits = tf.matmul(X,W) + b
    hypothesis = tf.nn.sigmoid(logits)

    cost_i = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,labels=Y)
    cost = tf.reduce_mean(cost_i)

    train = tf.train.GradientDescentOptimizer(learning_rate=0.2).minimize(cost)

    prediction = tf.cast(hypothesis > 0.5, dtype=tf.float32)
    correct_prediction = tf.equal(prediction, Y)
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, dtype=tf.float32))

    print("\n=====Processing=====")
    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        for step in range(10001):
```

```

        sess.run(train, feed_dict={X: pca_train_x, Y: train_y})
        if step % 1000 == 0:
            loss, acc = sess.run([cost, accuracy], feed_dict={X:
↳pca_train_x, Y: train_y})
            print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step,
↳loss, acc))

        train_acc = sess.run(accuracy, feed_dict={X: pca_train_x, Y: train_y})
        test_acc, test_predict, test_correct = sess.
↳run([accuracy, prediction, correct_prediction], feed_dict={X: pca_test_x, Y:
↳test_y})

        print("\n=====Results=====")
        print("PCA Model Prediction =", train_acc)
        print("PCA Test Prediction =", test_acc)

        return train_acc, test_acc

ann_slp_pca_train_acc, ann_slp_pca_test_acc = ann_slp_pca()

```

=====**Data Summary**=====

PCA Training Data : (398, 10)

PCA Testing Data : (171, 10)

=====**Processing**=====

Step:	0	Loss: 0.701	Acc: 54.52%
Step:	1000	Loss: 0.142	Acc: 96.23%
Step:	2000	Loss: 0.117	Acc: 96.98%
Step:	3000	Loss: 0.106	Acc: 97.49%
Step:	4000	Loss: 0.100	Acc: 97.74%
Step:	5000	Loss: 0.095	Acc: 97.74%
Step:	6000	Loss: 0.092	Acc: 97.74%
Step:	7000	Loss: 0.089	Acc: 97.74%
Step:	8000	Loss: 0.087	Acc: 97.99%
Step:	9000	Loss: 0.086	Acc: 97.99%
Step:	10000	Loss: 0.084	Acc: 97.99%

=====**Results**=====

PCA Model Prediction = 0.9799

PCA Test Prediction = 0.964912

```

[43]: def ann_mlp():
        print("=====Data Summary=====")
        print("Training Data :", train_x.shape)
        print("Testing Data :", test_x.shape)

        X = tf.placeholder(tf.float32, [None, 30])

```



```

Y = tf.placeholder(tf.float32, [None, 1])

# input
W1 = tf.Variable(tf.random_normal([30,60], seed=0), name='weight1')
b1 = tf.Variable(tf.random_normal([60], seed=0), name='bias1')
layer1 = tf.nn.sigmoid(tf.matmul(X,W1) + b1)

# hidden1
W2 = tf.Variable(tf.random_normal([60,60], seed=0), name='weight2')
b2 = tf.Variable(tf.random_normal([60], seed=0), name='bias2')
layer2 = tf.nn.sigmoid(tf.matmul(layer1,W2) + b2)

# hidden2
W3 = tf.Variable(tf.random_normal([60,90], seed=0), name='weight3')
b3 = tf.Variable(tf.random_normal([90], seed=0), name='bias3')
layer3 = tf.nn.sigmoid(tf.matmul(layer2,W3) + b3)

# output
W4 = tf.Variable(tf.random_normal([90,1], seed=0), name='weight4')
b4 = tf.Variable(tf.random_normal([1], seed=0), name='bias4')
logits = tf.matmul(layer3,W4) + b4
hypothesis = tf.nn.sigmoid(logits)

cost_i = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,labels=Y)
cost = tf.reduce_mean(cost_i)

train = tf.train.GradientDescentOptimizer(learning_rate=0.001).
↪ minimize(cost)

prediction = tf.cast(hypothesis > 0.5, dtype=tf.float32)
correct_prediction = tf.equal(prediction, Y)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, dtype=tf.float32))

print("\n=====Processing=====")
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: train_x, Y: train_y})
        if step % 1000 == 0:
            loss, acc = sess.run([cost, accuracy], feed_dict={X: train_x, Y:
↪ train_y})
            print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step,
↪ loss, acc))

    train_acc = sess.run(accuracy, feed_dict={X: train_x, Y: train_y})

```

```

        test_acc,test_predict,test_correct = sess.
↳run([accuracy,prediction,correct_prediction], feed_dict={X: test_x, Y:
↳test_y})

        print("\n=====Results=====")
        print("Model Prediction =", train_acc)
        print("Test Prediction =", test_acc)

        return train_acc,test_acc

ann_mlp_train_acc, ann_mlp_test_acc = ann_mlp()

```

=====Data Summary=====

Training Data : (398, 30)

Testing Data : (171, 30)

=====Processing=====

Step:	0	Loss: 2.073	Acc: 37.44%
Step:	1000	Loss: 0.335	Acc: 89.95%
Step:	2000	Loss: 0.266	Acc: 92.71%
Step:	3000	Loss: 0.226	Acc: 93.72%
Step:	4000	Loss: 0.199	Acc: 93.97%
Step:	5000	Loss: 0.180	Acc: 95.23%
Step:	6000	Loss: 0.166	Acc: 95.98%
Step:	7000	Loss: 0.155	Acc: 96.23%
Step:	8000	Loss: 0.147	Acc: 96.73%
Step:	9000	Loss: 0.139	Acc: 96.98%
Step:	10000	Loss: 0.133	Acc: 96.98%

=====Results=====

Model Prediction = 0.969849

Test Prediction = 0.929825

```

[44]: def ann_mlp_pca():
        sklearn_pca = sklearnPCA(n_components=10)

        print("=====Data Summary=====")
        pca_train_x = sklearn_pca.fit_transform(train_x)
        print("PCA Training Data :", pca_train_x.shape)

        pca_test_x = sklearn_pca.fit_transform(test_x)
        print("PCA Testing Data :", pca_test_x.shape)

        X = tf.placeholder(tf.float32, [None,10])
        Y = tf.placeholder(tf.float32, [None, 1])

        # input

```

```

W1 = tf.Variable(tf.random_normal([10,64], seed=0), name='weight1')
b1 = tf.Variable(tf.random_normal([64], seed=0), name='bias1')
layer1 = tf.nn.sigmoid(tf.matmul(X,W1) + b1)

# hidden1
W2 = tf.Variable(tf.random_normal([64,128], seed=0), name='weight2')
b2 = tf.Variable(tf.random_normal([128], seed=0), name='bias2')
layer2 = tf.nn.sigmoid(tf.matmul(layer1,W2) + b2)

# hidden2
W3 = tf.Variable(tf.random_normal([128,128], seed=0), name='weight3')
b3 = tf.Variable(tf.random_normal([128], seed=0), name='bias3')
layer3 = tf.nn.sigmoid(tf.matmul(layer2,W3) + b3)

# output
W4 = tf.Variable(tf.random_normal([128,1], seed=0), name='weight4')
b4 = tf.Variable(tf.random_normal([1], seed=0), name='bias4')
logits = tf.matmul(layer3,W4) + b4
hypothesis = tf.nn.sigmoid(logits)

cost_i = tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,labels=Y)
cost = tf.reduce_mean(cost_i)

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

prediction = tf.cast(hypothesis > 0.5, dtype=tf.float32)
correct_prediction = tf.equal(prediction, Y)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, dtype=tf.float32))

print("\n=====Processing=====")
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: pca_train_x, Y: train_y})
        if step % 1000 == 0:
            loss, acc = sess.run([cost, accuracy], feed_dict={X:
↪pca_train_x, Y: train_y})
            print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step,
↪loss, acc))

            train_acc = sess.run(accuracy, feed_dict={X: pca_train_x, Y: train_y})
            test_acc,test_predict,test_correct = sess.
↪run([accuracy,prediction,correct_prediction], feed_dict={X: pca_test_x, Y:
↪test_y})

    print("\n=====Results=====")
    print("PCA Model Prediction =", train_acc)

```

```

print("PCA Test Prediction =", test_acc)

return train_acc, test_acc

ann_mlp_pca_train_acc, ann_mlp_pca_test_acc = ann_mlp_pca()

```

====Data Summary=====

PCA Training Data : (398, 10)

PCA Testing Data : (171, 10)

====Processing=====

```

Step:    0      Loss: 2.958      Acc: 62.56%
Step:  1000     Loss: 0.109      Acc: 97.74%
Step:  2000     Loss: 0.086      Acc: 98.24%
Step:  3000     Loss: 0.075      Acc: 98.24%
Step:  4000     Loss: 0.068      Acc: 98.49%
Step:  5000     Loss: 0.063      Acc: 98.74%
Step:  6000     Loss: 0.059      Acc: 98.99%
Step:  7000     Loss: 0.055      Acc: 98.99%
Step:  8000     Loss: 0.053      Acc: 98.99%
Step:  9000     Loss: 0.051      Acc: 98.99%
Step: 10000     Loss: 0.049      Acc: 98.99%

```

====Results=====

PCA Model Prediction = 0.98995

PCA Test Prediction = 0.953216

```

[45]: sub = pd.DataFrame()
sub['id'] = test_id
sub['Predict_Type'] = test_predict.astype(int)
sub['Origin_Type'] = test_y
sub['Correct'] = test_correct
sub.head(10)

```

```

[45]:
      id  Predict_Type  Origin_Type  Correct
204  87930             1           1     True
70   859575            0           0     True
131   8670             0           0     True
431  907915             1           1     True
540  921385             1           1     True
567  927241             0           0     True
369  9012000            0           0     True
29   853201             0           0     True
81   8611161            0           1    False
477  911673             1           1     True

```

```

[46]: sub[['id', 'Predict_Type']].to_csv('submission.csv', index=False)

```