Introduction

Use DNN to predict the score of app in Google Play Store

Our goal is to use the other given data except the score to predict the score of application.Although in the given data,score Y is not a continuous value,we set the score Y a continuous value to do the machine learning as a matter of convenience.

First of all,we pre-process the data set to remove the duplicate data,empty data and the NaN value.

```
d = pd.read_csv('googleplaystore.csv')

d = d.astype(str)
d.drop_duplicates(subset='App', inplace=True)
d = d[d['Android Ver'] != np.nan]
d = d[d['Android Ver'] != 'NaN']
d = d[d['Installs'] != 'Free']
d = d[d['Installs'] != 'Paid']
d = d[d['Size'] != 'Varies with device']
d = d[d['Size'] != '']
d = d[d['Rating'] != 'NaN']
d = d[d['Rating'] != 'nan']
```

Then we transform the Rating,Installs,Price,Reviews to the numerical value,each maintains the same unit.

```
d['Installs'] = d['Installs'].apply(lambda x: x.replace('+', '')
d['Installs'] = d['Installs'].apply(lambda x: x.replace(',', '')
d['Installs'] = d['Installs'].apply(lambda x: int(x))
d['Size'] = d['Size'].apply(lambda x: str(x).replace('M', '') if
d['Size'] = d['Size'].apply(lambda x: str(x).replace(',', '') if
d['Size'] = d['Size'].apply(lambda x: float(str(x).replace('k',
d['Size'] = d['Size'].apply(lambda x: float(x))
d['Installs'] = d['Installs'].apply(lambda x: float(x))
d['Price'] = d['Price'].apply(lambda x: str(x).replace('$', '')
d['Price'] = d['Price'].apply(lambda x: float(x))
d['Reviews'] = d['Reviews'].apply(lambda x: int(x))
```

For the prediction of machine learning,we transform the character value to the numerical value and classify the same to one class,then we choose the enumerated index as the input of machine learning.

```
def get_index_value(name, x):
    a = d.groupby([name]).size().reset_index()
    for index, i in enumerate(a.values):
        if i[0] == str(x):
            return index
    return -1
d['Category Val'] = d['Category'].apply(lambda x: get_index_valu
d['Type Val'] = d['Type'].apply(lambda x: get_index_value('Type'
d['Content Rating Val'] = d['Content Rating'].apply(lambda x: ge
d['Android Ver Val'] = d['Android Ver'].apply(lambda x: get_inde
```

Then we construct the DNN model based on Keras to generate the mapping relationship.Considering that the amount of data in the data set is small,the complexity of the data is not high,we build two hidden layers,there are 64 neurons in each layer.We use the sigmoid function as the activation function of the neurons.We use the RMSprop as the learning optimization algorithm,it is a kind of adaptive learning rate method.We use mse as loss function.

```
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='sigmoid', input_shape=(2,)))
    model.add(Dense(64, activation='sigmoid'))
    model.add(Dense(1))

    rmsprop = RMSprop(lr=0.001)
    model.compile(optimizer=rmsprop, loss='mse', metrics=['mae']
    return model
```

We read out the cleaned and pre-processed data before,and divide the whole data into the traning set and test set.There are 6000 data in the training set and 1027 data int the test set.Due to the large scale of magnitude between each class of training data,so we normalize the data at the same time.we use the standard deviation standardization by minusing the average value and divding by the standard deviation to make the processed data meet the normal distribution.

```
df = pd.read_csv('googleplaystore_clean_val.csv')
a = np.copy(df.values)
a = np.delete(a, 0, axis=1)
y = np.copy(a[:, 0])
x = np.delete(a, 0, axis=1)
x_train = x[0:6000]
y_train = y[0:6000]
x_test = x[6000:7027]
y_test = y[6000:7027]

mean = x_train.mean(axis=0)
x_train -= mean
std = x_train.std(axis=0)
x_train /= std
x_test -= mean
x_test /= std
```

In the end,it is training and the preservation of the training model.We read the model preserved and use the test set to predict and evaluate the result.

```
is_train = False
    if is_train:
        model = build_model()
        model.fit(x_train, y_train, epochs=180, batch_size=16, v
        model.save('dnn_2_4_col.h5')
    else:
        model = keras.models.load_model('dnn_2_4_col.h5')
        y_ = model.predict(x_test).reshape(-1)
        # y_a = np.around(y_, 1)
        count = 0
        for i in range(len(y_)):
            if abs(y_[i] - y_test[i]) <= 0.2:
                count += 1
        print('accuracy: {}%'.format(np.around(count / len(y_) *
        print('actual - mean: {}'.format(y_test.mean()))
        print('predict - mean: {}'.format(y_.mean()))

        print(y_test.min())
        print(y_test.max())

        top = y_test + 0.2
        bottom = y_test - 0.2

        plt.scatter(y_, y_test, s=5)
        plt.plot(y_test, y_test, 'r')
        plt.plot(y_test, top, alpha=0.5)
```

```
        plt.plot(y_test, bottom, alpha=0.5)
        plt.xlim([4.0, 5])
        plt.xlabel('actual')
        plt.ylabel('predict')
        plt.show()
```

The average score predicted by the model is 4.21,the actual average is 4.11.In the situation allowing error plus or minus 0.5,the accuracy of the prediction is 69.43%.

After using the DNN to predict the score of applications in Google Play Store,we hope DNN can learn the influences to the score in some less obvious dimensions including Size,Category and so on.The results of the predications are not particularly satisfactory,the possible reason is the correlation between the score and these dimensions.