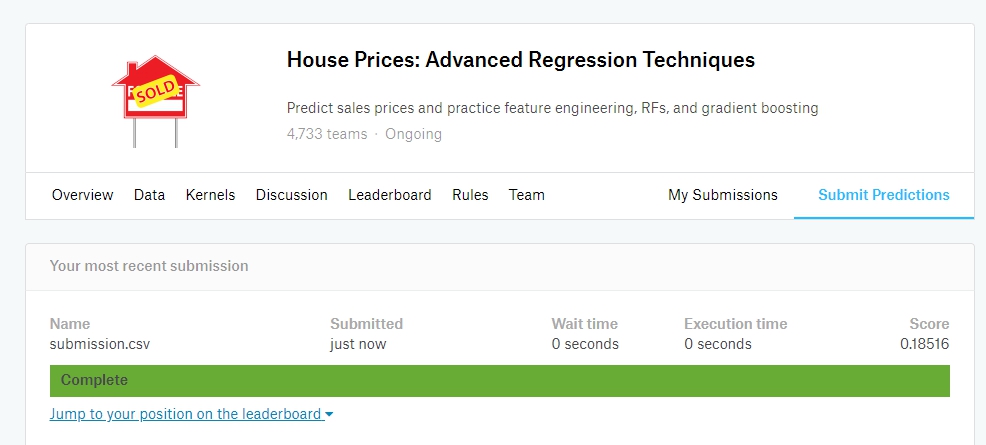
1041501\_hw2

黃威竣

建置環境為Anaconda Nacigatot Jupyter Notebook



下圖為kaggle上的評分



我是使用Gradient decent的方法去找出權重，以及刪除某些很明顯無關的特徵以及補Nan

以下為程式碼 有註解

使用到的函式庫有 pandas,numpy , sklearn.preprocessing

#!/usr/bin/env python

# coding: utf-8

# In[68]:

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import random

import math

from sklearn.preprocessing import StandardScaler

train\_csv = pd.read\_csv('train.csv') #read CSV檔 file

test\_csv = pd.read\_csv('test.csv')

train\_df = pd.DataFrame(train\_csv) #to panda dataFrame

test\_df = pd.DataFrame(test\_csv)

train\_df = train\_df.drop('Alley',1) ##以下移除的這4筆column很明顯與資料結果無關係

train\_df = train\_df.drop('PoolQC',1)

train\_df = train\_df.drop('MiscFeature',1)

train\_df = train\_df.drop('Fence',1)

train\_df["GarageYrBlt"].fillna(0,inplace = True)

train\_df["Utilities"].fillna("None",inplace = True) #把這幾筆Column的Nan補上None值

train\_df["Exterior1st"].fillna("None",inplace = True)

train\_df["Exterior2nd"].fillna("None",inplace = True)

train\_df["FireplaceQu"].fillna("None",inplace = True)

test\_df = test\_df.drop('Alley',1) ##以下移除的這4筆column很明顯與資料結果無關係

test\_df = test\_df.drop('PoolQC',1)

test\_df = test\_df.drop('MiscFeature',1)

test\_df = test\_df.drop('Fence',1)

test\_df["GarageYrBlt"].fillna(0,inplace = True)

test\_df["Utilities"].fillna("None",inplace = True) #把這幾筆Column的Nan補上None值

test\_df["Exterior1st"].fillna("None",inplace = True)

test\_df["Exterior2nd"].fillna("None",inplace = True)

test\_df["FireplaceQu"].fillna("None",inplace = True)

test\_df = test\_df.fillna(value=0)

train\_df = train\_df.fillna(value=0)

y\_train = train\_df["SalePrice"].values #轉成np

t = y\_train

X\_train = train\_df = train\_df.drop('SalePrice', 1)

for col\_name in X\_train.columns:

if(X\_train[col\_name].dtype == 'object'):

X\_train[col\_name]= X\_train[col\_name].astype('category')

X\_train[col\_name] = X\_train[col\_name].cat.codes

for col\_name in test\_df.columns:

if(test\_df[col\_name].dtype == 'object'):

test\_df[col\_name]= test\_df[col\_name].astype('category')

test\_df[col\_name] = test\_df[col\_name].cat.codes

X\_test = test\_df.values #轉成np

X\_train = X\_train.values

# In[ ]:

# In[69]:

## 先讓X與y去作標準化 使用StandardScaler()

y\_train = y\_train.reshape(1460,1)

SC\_X = StandardScaler()

SC\_y = StandardScaler()

X\_train = SC\_X.fit\_transform(X\_train) ##原本是int標準化會有小數點自然會有數字會遺失出現警告

y\_train = SC\_y.fit\_transform(y\_train)

X\_test = SC\_X.transform(X\_test)

for i in range (len(X\_train)):

X\_train[i][0] = 1

for i in range (len(X\_test)):

X\_test[i][0] = 1

# ### 我寫的Gradient\_desent function theta為要找出的角度,alpha為learning rate, num\_iters為次數

# In[63]:

def gradient\_Descent(X, y, theta, alpha, num\_iters):

m = len(y)

n = len(theta)

tmp = theta

theta.reshape(n,1)

y\_pred = np.dot(X,theta)

for i in range(num\_iters):

for j in range(n):

tmp[j] = tmp[j] - alpha \* (1/m)\*sum((y\_pred - y)\*X[:,j]) ##計算theta最關鍵的就是這一行，也是微分過後的公式帶進去一直算

y\_pred = np.dot(X,theta)

cost = (1.0 / (2.0 \* m)) \* sum((y\_pred-y)\*(y\_pred-y)) #這邊的Y都是標準化過後的Y

print ("iteration {}, cost {}".format(i,cost))

return theta

# In[ ]:

# In[64]:

y\_train = y\_train.flatten() #我丟進去的y要是一維,原本二維用這個涵式降維

theta = np.random.rand(len(X\_train[0])) #theta我採用隨機的數值

gradient\_Descent(X\_train,y\_train,theta,0.1,500) #呼叫Gradient\_descent #看紀錄大致上500次就差不多了 alpha=0.1的原因是只要讓他可以穩定就好

# In[65]:

ANS = np.dot(X\_test,theta) #找出來的theta值與需要預測的資料內積

ANS\_inv = ANS.reshape(1459,1) ## 由於前面的"Price"有作標準化，在這邊要把預測出的結果給inverse回去

ANS\_inv = SC\_y.inverse\_transform((ANS\_inv))

ANS\_inv = ANS\_inv.flatten()

Id = np.zeros(len(X\_test))

for i in range(len(X\_test)): ##寫檔用的Id

Id[i]=i+1461

Id = Id.astype(np.int32)

f = open("submission.csv", "w") ##寫檔

f.write("{},{}\n".format("Id", "SalePrice"))

for x in zip(Id, ANS\_inv):

f.write("{},{}\n".format(x[0], x[1]))

f.close()

# In[67]:

print(ANS\_inv)

# In[ ]: