# 机器学习第二次作业

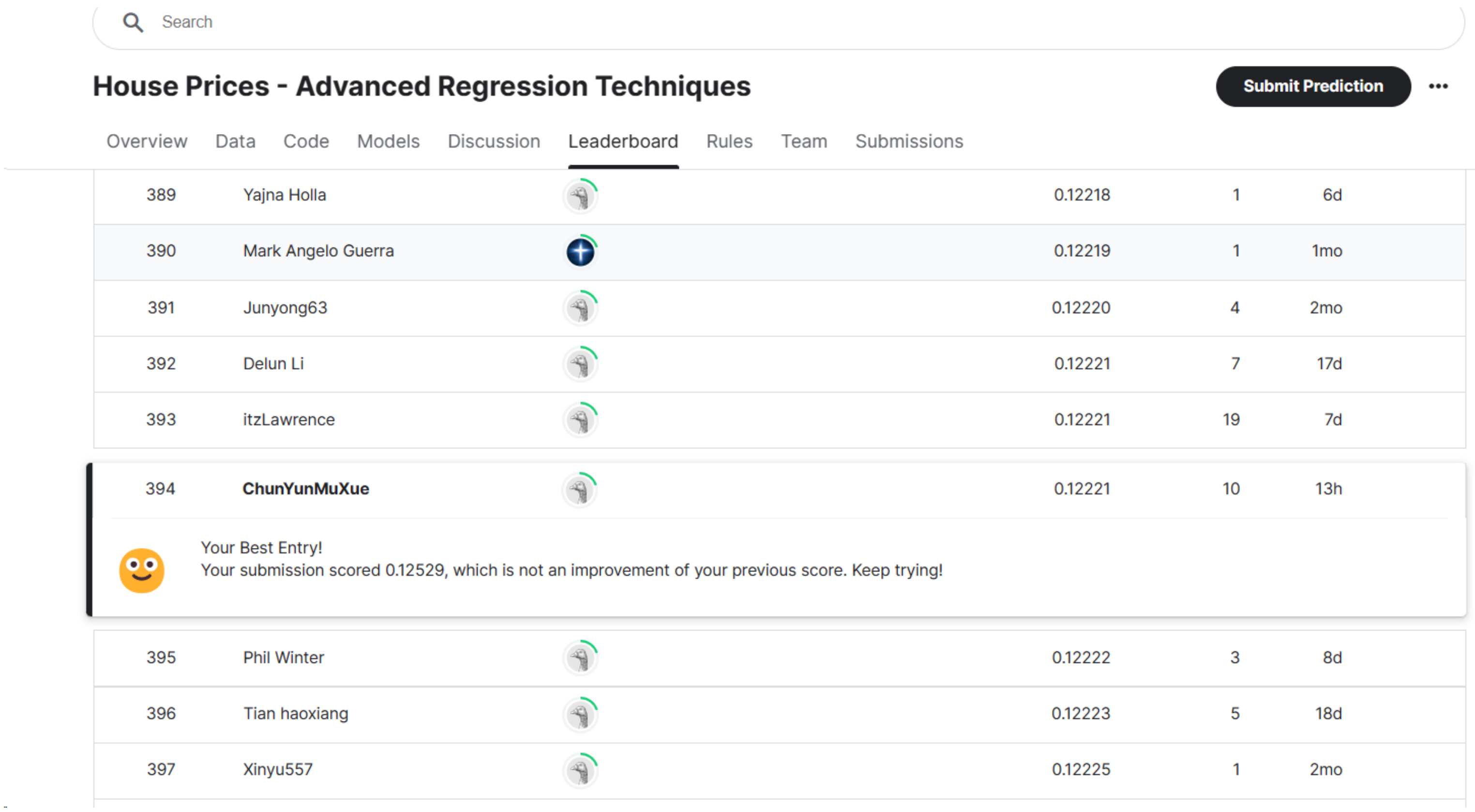
- ☑ 1.对比一下GoogLeNet的Inception V1V2V3V4四个版本的不同;
- ☑ 2.用自己的线性回归模型跑一下【波士顿房价数据集】;
  Kaggle地址: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques
  要求呈现完整的代码和结果;
- ☑ 3.对于人脸防伪任务来说，recall值和precision值应该怎么组合，说出想法并解释原因。
- ☑ 4.调研一下recall值和precision值的不同组合，什么时候关注precision值多一些，什么时候关注recall值多一些?
- ☑ 5.对于目标检测任务来说，如果我们不关注边界框的proposal，模型的输入、输出、训练会有什么问题，有哪些弊端?

## 1.对比一下GoogLeNet的Inception V1V2V3V4四个版本的不同;

作业为GoogLeNet.pdf 源码为GoogLeNet.md 原markdown的图片库为GoogLeNet.assets

## 2.用自己的线性回归模型跑一下【波士顿房价数据集】

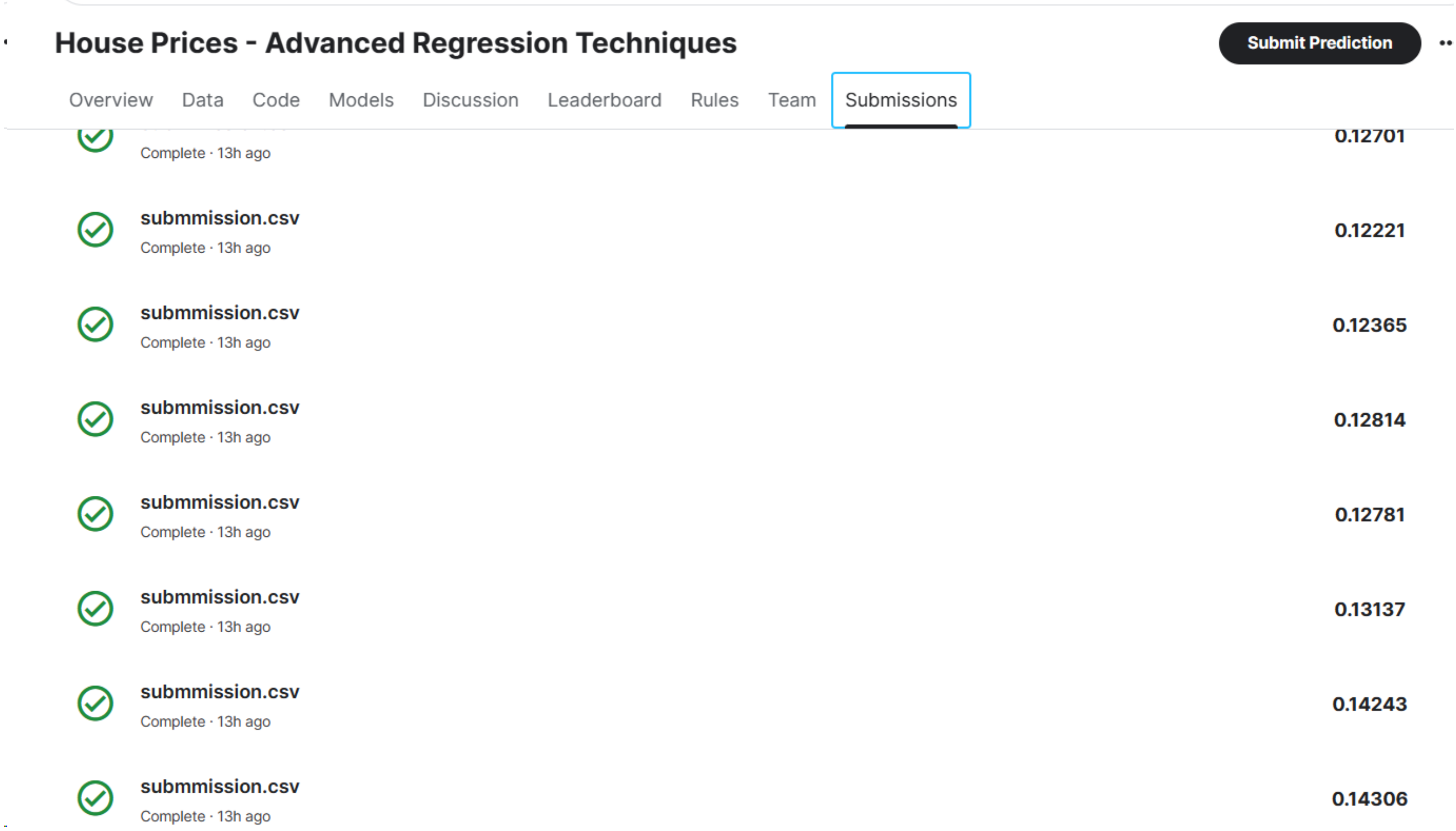
结果:



House Prices - Advanced Regression Techniques

| | | | | |
|---|---|---|---|---|
| 389 | Yajna Holla | 0.12218 | 1 | 6d |
| 390 | Mark Angelo Guerra | 0.12219 | 1 | 1mo |
| 391 | Junyong63 | 0.12220 | 4 | 2mo |
| 392 | Delun Li | 0.12221 | 7 | 17d |
| 393 | itzLawrence | 0.12221 | 19 | 7d |
| 394 | **ChunYunMuXue** | 0.12221 | 10 | 13h |
| 395 | Phil Winter | 0.12222 | 3 | 8d |
| 396 | Tian haoxiang | 0.12223 | 5 | 18d |
| 397 | Xinyu557 | 0.12225 | 1 | 2mo |

Your Best Entry!
Your submission scored 0.12529, which is not an improvement of your previous score. Keep trying!



House Prices - Advanced Regression Techniques

| | | |
|---|---|---|
| | Complete · 13h ago | 0.12701 |
| submmission.csv | Complete · 13h ago | 0.12221 |
| submmission.csv | Complete · 13h ago | 0.12365 |
| submmission.csv | Complete · 13h ago | 0.12814 |
| submmission.csv | Complete · 13h ago | 0.12781 |
| submmission.csv | Complete · 13h ago | 0.13137 |
| submmission.csv | Complete · 13h ago | 0.14243 |
| submmission.csv | Complete · 13h ago | 0.14306 |

## 代码

```
import torch
import torch.nn as nn
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from torch.utils.data import Dataset
from torch.utils.data import DataLoader

train_data = pd.read_csv(open("D:/article and study/study/Dian_serach_about/machine_learning_Dian/home_work_week4/house_prices/house-prices-advanced-regression-techniques/train.csv"))
test_data = pd.read_csv(open("D:/article and study/study/Dian_serach_about/machine_learning_Dian/home_work_week4/house_prices/house-prices-advanced-regression-techniques/test.csv"))

F1 = plt.figure()
abs(train_data.corr(numeric_only = True)['SalePrice']).sort_values(ascending=False).plot.bar()
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 20)
plt.tight_layout()


# 删除离群点

train_data = train_data.drop(train_data[(train_data['OverallQual']<5) & (train_data['SalePrice']>200000)].index)
train_data = train_data.drop(train_data[(train_data['GrLivArea']>4000) & (train_data['SalePrice']<300000)].index)
train_data = train_data.drop(train_data[(train_data['YearBuilt']<1900) & (train_data['SalePrice']>400000)].index)
train_data = train_data.drop(train_data[(train_data['TotalBsmtSF']>6000) & (train_data['SalePrice']<200000)].index)

all_features = pd.concat((train_data.iloc[:,1:-1],test_data.iloc[:,1:]))

numeric_features = all_features.dtypes[all_features.dtypes != 'object'].index

all_features[numeric_features]=all_features[numeric_features].apply(lambda x:(x-x.mean())/(x.std()))# 正态分布归一化

all_features[numeric_features]=all_features[numeric_features].fillna(0)
```

```
all_features = pd.get_dummies(all_features,dummy_na = True,dtype = float)

#-------------------------------------------------------------------------------------------
-------------------- Value
batch_size = 100
num_workers = 3
lr = 0.095
weight_decay = 350
Epoch = 200
test_size = 40
#-------------------------------------------------------------------------------------------
-------------------- data clear

n = train_data.shape[0]

train_fetures = torch.tensor(all_features[:n].values,dtype=torch.float)[:-test_size]
train_res = torch.tensor(train_data["SalePrice"].values,dtype=torch.float).view(-1,1)[:-
test_size]

test_fetures = torch.tensor(all_features[:n].values,dtype=torch.float)[-test_size:]
test_res = torch.tensor(train_data["SalePrice"].values,dtype=torch.float).view(-1,1)[-
test_size:]

Ans_fetures = torch.tensor(all_features[n:].values,dtype=torch.float)

class Data(Dataset):
    def __init__(self,fetures,res):
        self.x_data = fetures
        self.y_data = res
        self.len = fetures.shape[0]
        np.savetxt("house_prices/house-prices-advanced-regression-
techniques/train_del.csv",self.x_data,delimiter = ',')
        np.savetxt("house_prices/house-prices-advanced-regression-
techniques/train_ans.csv",self.y_data,delimiter = ',')
    def __getitem__(self,index):
        return self.x_data[index],self.y_data[index]
    def __len__(self):
        return self.len

dataset = Data(train_fetures,train_res)
tdataset = Data(test_fetures,test_res)
```

```
train_loader = DataLoader(dataset = dataset,batch_size = batch_size,shuffle = True)
test_loader = DataLoader(dataset = tdataset,batch_size = batch_size,shuffle = False)

Len = test_fetures.shape[1]
H = test_fetures.shape[0]

# #-----------------------------------------------------------------------------------------
----------------------- data loader

class Model(torch.nn.Module):
    def __init__(self):
        super(Model,self).__init__()
        self.linear1 = torch.nn.Linear(Len,256)
        self.linear2 = torch.nn.Linear(256,1)
        self.active = torch.nn.ReLU()

    def forward(self,x):
        x = self.active(self.linear1(x))
        x = self.active(self.linear2(x))
        return x

model = Model().float()
criterion = torch.nn.MSELoss(reduction = "mean")
optimizer=torch.optim.Adam(params = model.parameters(),lr = lr,weight_decay = weight_decay)
#-------------------------------------------------------------------------------------------
---------------------- Model

TRAIN_LS = []
TEST_LS = []
X_LS = []

def log_MSE(net,features,labels):
    with torch.no_grad():
        clipped_preds = torch.max(net(features),torch.tensor(1.0))
        rmse = torch.sqrt(criterion(clipped_preds.log(),labels.log()))
    return rmse.item()

def train(epoch):
    running_loss = 0.0
    for batch_idx,data in enumerate(train_loader,0):
        inputs,target = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs,target)
        loss.backward()
        optimizer.step()
        running_loss += log_MSE(model,inputs,target)
    # print('Train : [%d] Loss : %.3f' % (epoch + 1,running_loss / (H / batch_size)))
    X_LS.append(epoch)
    TRAIN_LS.append(running_loss / (H / batch_size))

# ------------------------------------------------------------------------------------------
-----------------------Train

def test(epoch):
    running_loss = 0.0
    for batch_idx,data in enumerate(test_loader,0):
        inputs,target = data
        running_loss += log_MSE(model,inputs,target)
    # print('Test : [%d] Loss : %.3f' % (epoch + 1,running_loss / (test_size / batch_size)))
    TEST_LS.append(running_loss / (test_size / batch_size))

# ------------------------------------------------------------------------------------------
-----------------------Test

def pre():
    Ans = []
    with torch.no_grad():
        for row in range(len(Ans_fetures)):
            inputs = Ans_fetures[row]
            ans_pre = model(inputs).item()
            Ans.append([1461 + row,ans_pre])
    names = ['Id','SalePrice']
    Ans = pd.DataFrame(columns = names,data = Ans)
    Ans.to_csv('house_prices/house-prices-advanced-regression-
techniques/submmission.csv',index = None)
    # print(Ans)

# ------------------------------------------------------------------------------------------
-----------------------Ans
if __name__ == '__main__':
    for epoch in range(Epoch):
        train(epoch)
        test(epoch)
    F = plt.figure()
    plt.plot(X_LS,TRAIN_LS,'royalblue');
    plt.plot(X_LS,TEST_LS,'darkorange');
    plt.ylabel('RMSELoss')
    plt.xlabel('Epoch')
    plt.xticks(fontsize = 30)
    plt.yticks(fontsize = 20)
    # plt.show()
    pre()
# ------------------------------------------------------------------------------------------
-----------------------Main
```

源代码在 house_price 文件夹中的code.py

数据清洗部分参考了kaggle比赛：房价预测（排名前4%）_房价预测kaggle 前4%-CSDN博客

### 3.对于人脸防伪任务来说，recall值和precision值应该怎么组合，说出想法并解释原因。

### 4.调研一下recall值和precision值的不同组合，什么时候关注precision值多一些，什么时候关注recall值多一些?

### 5.对于目标检测任务来说，如果我们不关注边界框的proposal，模型的输入、输出、训练会有什么问题，有哪些弊端?

这三个任务一起位于 Recall and Precision.pdf 中，其中 Recall and Precision.md为markdown源文件，Recall and Precision.assert为用到的图片集