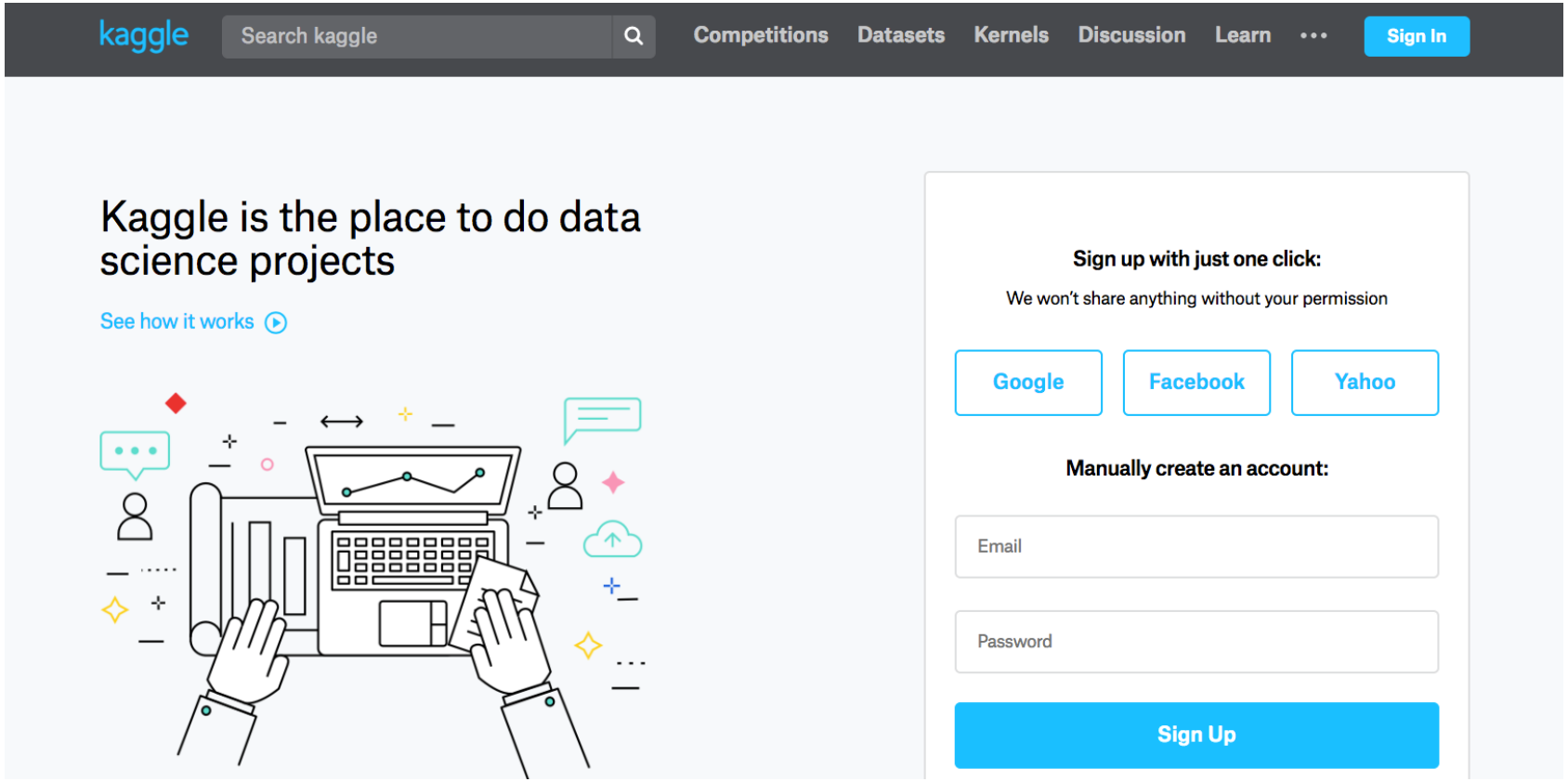
Predicting House Prices on Kaggle

Eng Teong Cheah

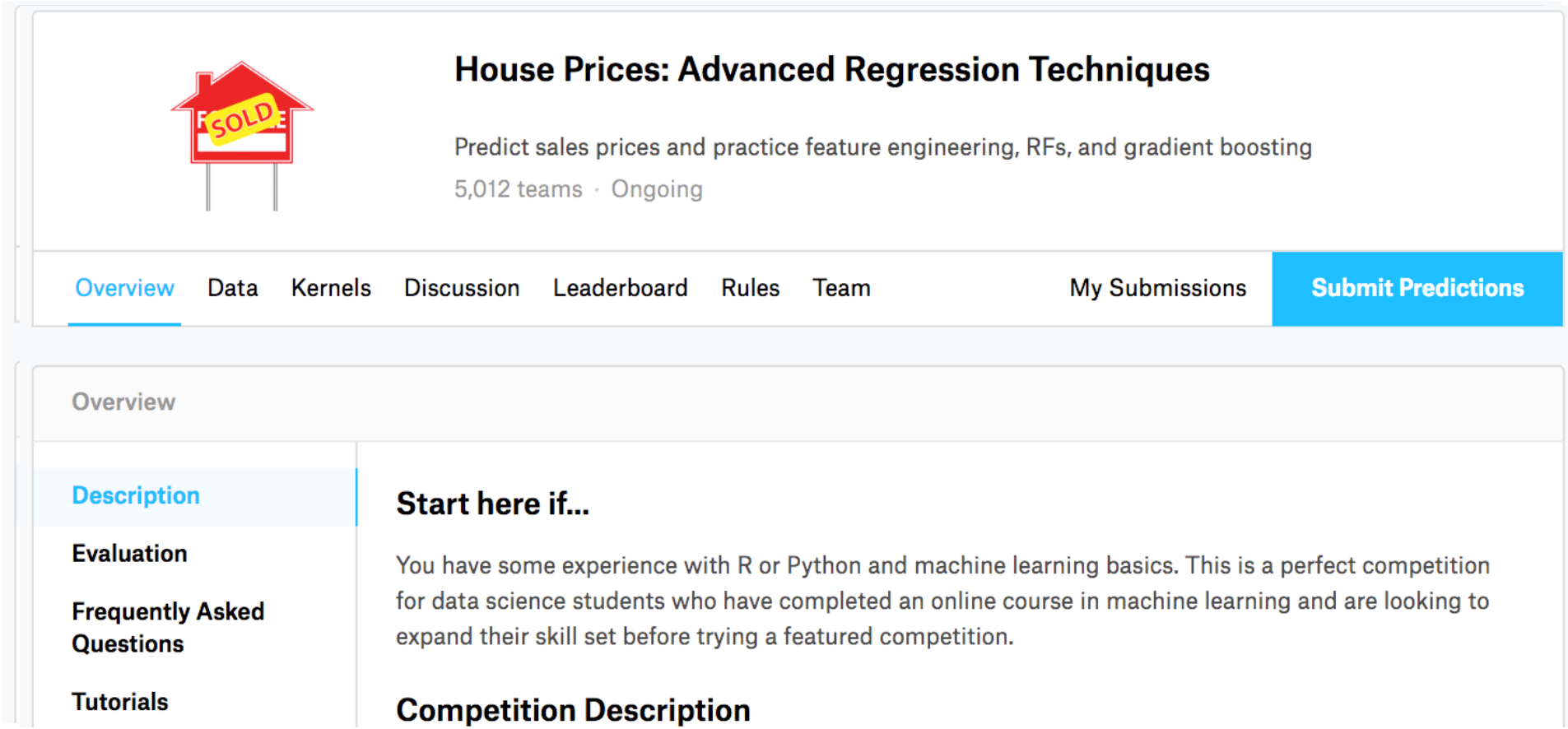
Contents

Predicting House Prices on Kaggle

Kaggle


kaggle
Search kaggle
Competitions
Datasets
Kernels
Discussion
Learn
Sign In
Kaggle is the place to do data science projects
See how it works
Sign up with just one click:
We won't share anything without your permission
Google
Facebook
Yahoo
Manually create an account:
Email
Password
Sign Up

Kaggle



https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Accessing and Reading Data Sets

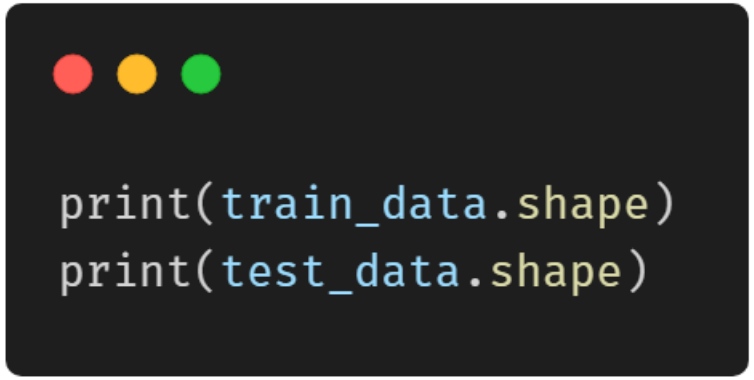
```
# If pandas is not installed, please uncomment the following line:
# !pip install pandas

%matplotlib inline
import d2l
from mxnet import autograd, gluon, init, nd
from mxnet.gluon import data as gdata, loss as gloss, nn
import numpy as np
import pandas as pd
```

Accessing and Reading Data Sets

```
train_data = pd.read_csv('../data/kaggle_house_pred_train.csv')
test_data = pd.read_csv('../data/kaggle_house_pred_test.csv')
```

Accessing and Reading Data Sets



Accessing and Reading Data Sets

```
train_data.iloc[0:4, [0, 1, 2, 3, -3, -2,
-1]]
```

	Id	MSSubClass	MSZoning	LotFrontage	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	WD	Normal	208500
1	2	20	RL	80.0	WD	Normal	181500
2	3	60	RL	68.0	WD	Normal	223500
3	4	70	RL	60.0	WD	Abnorml	140000

Accessing and Reading Data Sets

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

Data Preprocessing

```
numeric_features = all_features.dtypes[all_features.dtypes ≠
all_features[numeric_features] = all_features[numeric_features].apply(
    lambda x: (x - x.mean()) / (x.std()))
# After standardizing the data all means vanish, hence we can set missing
# values to 0
all_features[numeric_features] = all_features[numeric_features].fillna(0)
```

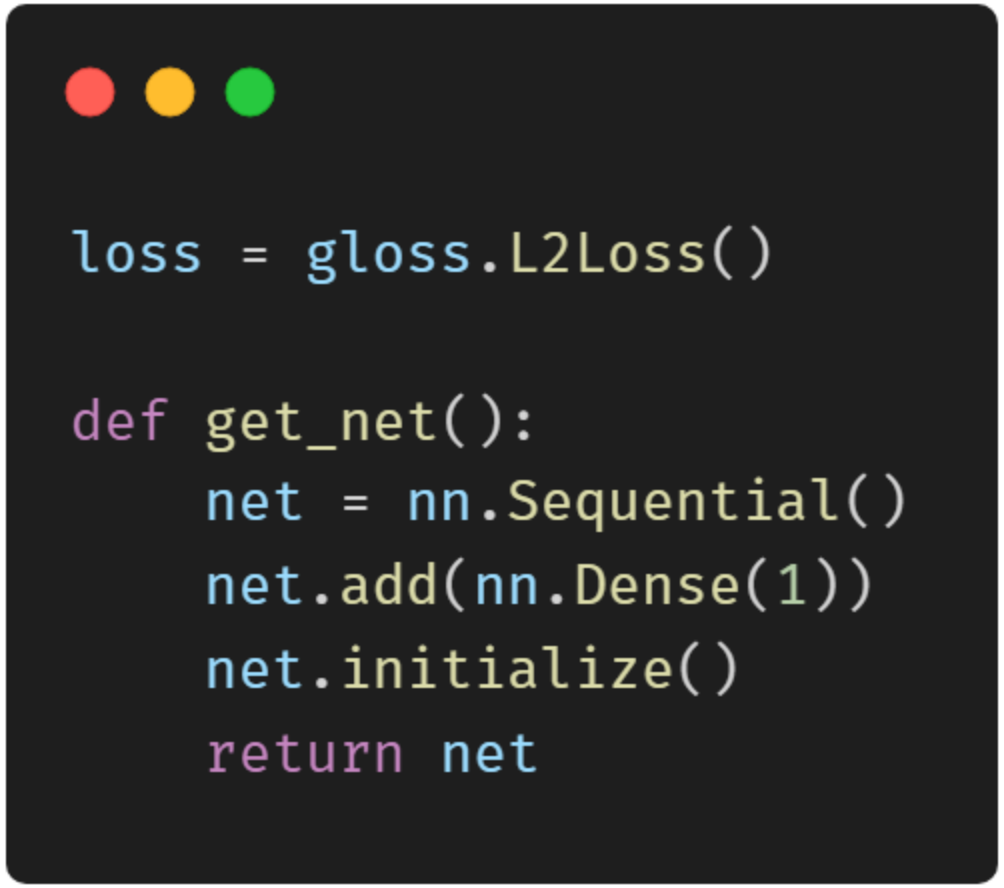
Data Preprocessing

```
# Dummy_na=True refers to a missing value being a legal eigenvalue, and
# creates an indicative feature for it
all_features = pd.get_dummies(all_features, dummy_na=True)
all_features.shape
```

Data Preprocessing

```
n_train = train_data.shape[0]
train_features = nd.array(all_features[:n_train].values)
test_features = nd.array(all_features[n_train:].values)
train_labels = nd.array(train_data.SalePrice.values).reshape((-1,
1))
```

Training



```
loss = gloss.L2Loss()

def get_net():
    net = nn.Sequential()
    net.add(nn.Dense(1))
    net.initialize()
    return net
```

Training

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i) - \log(\hat{y}_i))^2}$$

```
def log_rmse(net, features, labels):
    # To further stabilize the value when the logarithm is taken, set the
    # value less than 1 as 1
    clipped_preds = nd.clip(net(features), 1, float('inf'))
    rmse = nd.sqrt(2 * loss(clipped_preds.log(), labels.log()).mean())
    return rmse.asscalar()
```

Training

```
def train(net, train_features, train_labels, test_features, test_labels,
          num_epochs, learning_rate, weight_decay, batch_size):
    train_ls, test_ls = [], []
    train_iter = gdata.DataLoader(gdata.ArrayDataset(
        train_features, train_labels), batch_size, shuffle=True)
    # The Adam optimization algorithm is used here
    trainer = gluon.Trainer(net.collect_params(), 'adam', {
        'learning_rate': learning_rate, 'wd': weight_decay})
    for epoch in range(num_epochs):
        for X, y in train_iter:
            with autograd.record():
                l = loss(net(X), y)
            l.backward()
            trainer.step(batch_size)
        train_ls.append(log_rmse(net, train_features, train_labels))
        if test_labels is not None:
            test_ls.append(log_rmse(net, test_features, test_labels))
    return train_ls, test_ls
```

k-Fold Cross-Validation

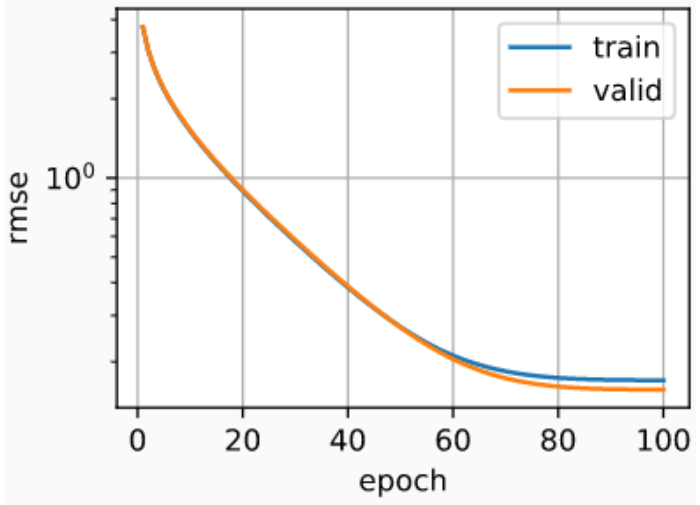
```
def get_k_fold_data(k, i, X, y):
    assert k > 1
    fold_size = X.shape[0] // k
    X_train, y_train = None, None
    for j in range(k):
        idx = slice(j * fold_size, (j + 1) *
fold_size)part, y_part = X[idx, :], y[idx]
        if j == i:
            X_valid, y_valid = X_part, y_part
        elif X_train is None:
            X_train, y_train = X_part, y_part
        else:
            X_train = nd.concat(X_train, X_part,
dim=0)       y_train = nd.concat(y_train, y_part,
dim=0)return X_train, y_train, X_valid, y_valid
```

k-Fold Cross-Validation

```
def k_fold(k, X_train, y_train, num_epochs,
           learning_rate, weight_decay, batch_size):
    train_l_sum, valid_l_sum = 0, 0
    for i in range(k):
        data = get_k_fold_data(k, i, X_train, y_train)
        net = get_net()
        train_ls, valid_ls = train(net, *data, num_epochs, learning_rate,
                                   weight_decay, batch_size)
        train_l_sum += train_ls[-1]
        valid_l_sum += valid_ls[-1]
        if i == 0:
            d2l.plot(list(range(1, num_epochs+1)), [train_ls, valid_ls],
                     xlabel='epoch', ylabel='rmse',
                     legend=['train', 'valid'], yscale='log')
        print('fold %d, train rmse: %f, valid rmse: %f' % (
            i, train_ls[-1], valid_ls[-1]))
    return train_l_sum / k, valid_l_sum / k
```

Model Selection

```
k, num_epochs, lr, weight_decay, batch_size = 5, 100, 5, 0, 64
train_l, valid_l = k_fold(k, train_features, train_labels, num_epochs,
lr,                       weight_decay, batch_size)
print('%d-fold validation: avg train rmse: %f, avg valid rmse: %f'
      % (k, train_l, valid_l))
```

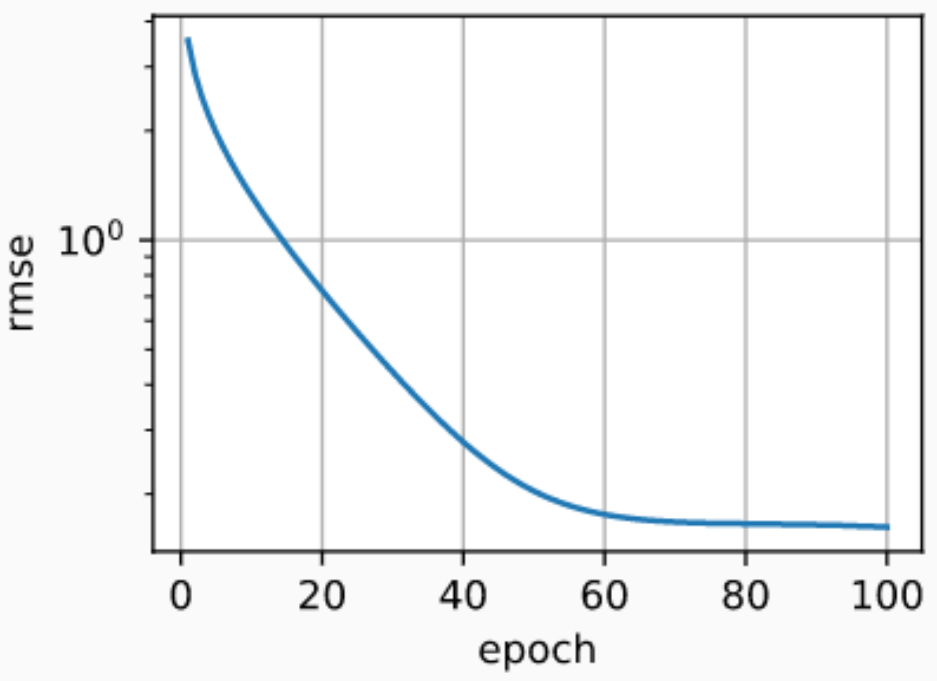


Predict and Submit

```
def train_and_pred(train_features, test_features, train_labels, test_data,
                   num_epochs, lr, weight_decay, batch_size):
    net = get_net()
    train_ls, _ = train(net, train_features, train_labels, None, None,
                        num_epochs, lr, weight_decay, batch_size)
    d2l.semilogy(range(1, num_epochs + 1), train_ls, 'epochs', 'rmse')
    print('train rmse %f' % train_ls[-1])
    preds = net(test_features).asnumpy()
    test_data['SalePrice'] = pd.Series(preds.reshape(1, -1)[0])
    submission = pd.concat([test_data['Id'], test_data['SalePrice']],
axis=1)
    submission.to_csv('submission.csv', index=False)
```

Predict and Submit

```
train_and_pred(train_features, test_features, train_labels, test_data,
               num_epochs, lr, weight_decay, batch_size)
```



Predict and Submit

Step 1
Upload submission file

Upload Submission File

File Format
Your submission should be in CSV format. You can upload this in a zip/gz/rar/7z archive, if you prefer.

Number of Predictions
We expect the solution file to have 1459 prediction rows. This file should have a header row. Please see sample submission file on the data page.

Step 2
Describe submission

Styling with Markdown supported

Briefly describe your submission.

Make Submission

Thank You !

Does anyone have any questions?

Twitter: @walkercet
Blog: https://ceteongvanness.wordpress.com

Resources

Dive into Deep Learning