```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from google.colab import files


from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error
import time
import scipy.stats as stats


# project done in Google Co-lab to take advantage of the GPU Computational power
!pip install -U -q PyDrive

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# 1. Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# PyDrive reference:
# https://googledrive.github.io/PyDrive/docs/build/html/index.html

# 2. Create & upload a file text file.
# uploaded = drive.CreateFile({'title': 'Sample upload.txt'})
# uploaded.SetContentString('Sample upload file content')
# uploaded.Upload()
# print('Uploaded file with ID {}'.format(uploaded.get('id')))

# # 3. Load a file by ID and print its contents.
# downloaded = drive.CreateFile({'id': uploaded.get('id')})
# print('Downloaded content "{}"'.format(downloaded.GetContentString()))


import os
fList = drive.ListFile({'q':"'1kPNjhSKgaQQY-AYX3EufYTO8DdVW3D65' in parents"}).GetList()
local_download_path=os.path.expanduser('~')
for f in fList:
  # 3. Create & download by id.
  print('title: %s, id: %s' % (f['title'], f['id']))
  fname = os.path.join(local_download_path, f['title'])
  print('downloading to {}'.format(fname))
  f_ = drive.CreateFile({'id': f['id']})
  f_.GetContentFile(fname)


with open(fname, 'r') as f:
  print(f.read())
```

```
    title: dataset_encoded, id: 1e4o8l-Cw2EWS-VpCDoo4uscyEG48ZrL4
    downloading to /content/dataset_encoded
    title: dataTestE, id: 1fhdVakV9zwEBhgxH3hU6fJQugI0eQRve
    downloading to /content/dataTestE
    title: pml_test_features.csv, id: 1_DdNm0SCrGgaUDEc9QaRD0bKKjsz-bHO
    downloading to /content/pml_test_features.csv
    IOPub data rate exceeded.
    The notebook server will temporarily stop sending output
    to the client in order to avoid crashing it.
    To change this limit, set the config variable
    `--NotebookApp.iopub_data_rate_limit`.
```

```
dataset_encoded = np.fromfile('/content/dataset_encoded')
```

```
    NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
```

```
dataTestE = np.fromfile('/content/dataTestE')
```

```
dataset_encoded=dataset_encoded.reshape(131822,1154)
dataTestE = dataTestE.reshape(56496,1153)
```

```
rows, columns = dataset_encoded.shape
data = dataset_encoded[:,0:(columns-1)] # remove loss column
loss = dataset_encoded[:,(columns-1)] # loss column
```

```
seed = 2016
model = XGBRegressor(seed=2016)

max_depth = range(3,10)

tuned_parameters=[{'max_depth': max_depth}]
n_folds = 3
clf = GridSearchCV(model,tuned_parameters, cv = n_folds,
                   scoring='neg_mean_absolute_error',
                   n_jobs=-1,verbose=1)
```

```
model = XGBRegressor(seed=2016,max_depth=5)
```

```
model.fit(data,loss,eval_metric='mae')
```

```
    XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
           colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
           max_depth=5, min_child_weight=1, missing=None, n_estimators=100,
           n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
           reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=2016,
           silent=True, subsample=1)
```

```
predictions = np.expm1(model.predict(dataTestE))
```

```
predLoss = pd.Series(predictions,name='loss')
submission  = pd.concat([df_test['id'],predLoss],axis=1)
```

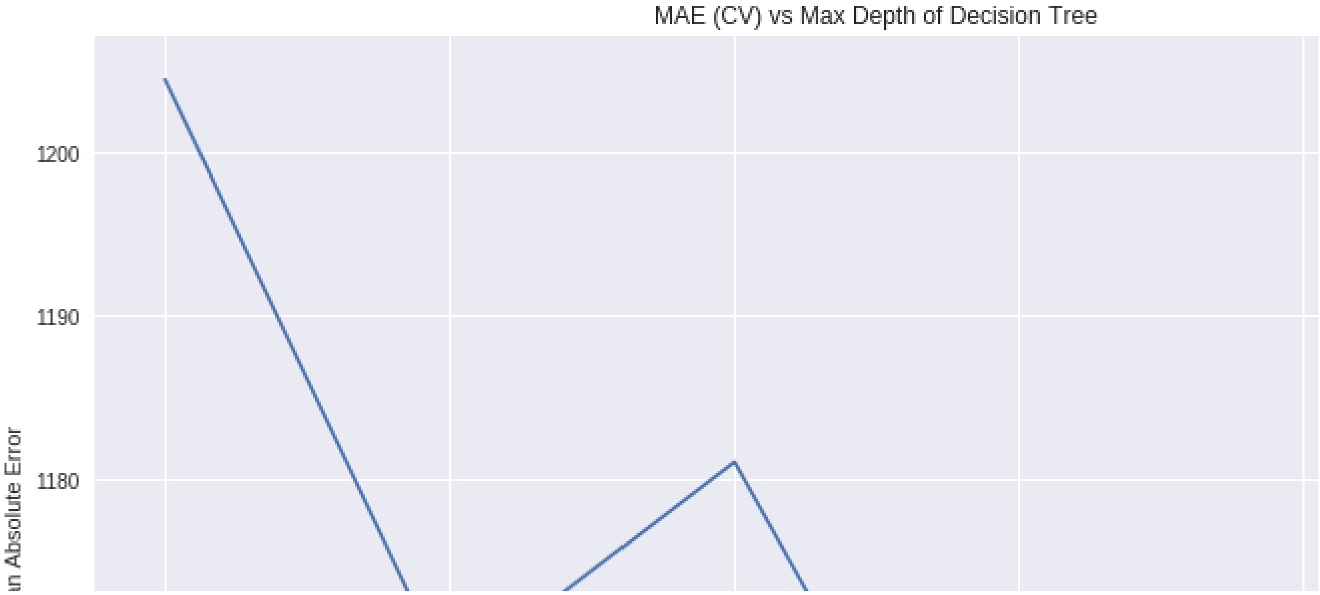
```
df_test = pd.read_csv('/content/pml_test_features.csv')
```

```
submission.to_csv('submission-XGBOOST3-5Depth.csv',index=False)
files.download('submission-XGBOOST3-5Depth.csv')
```

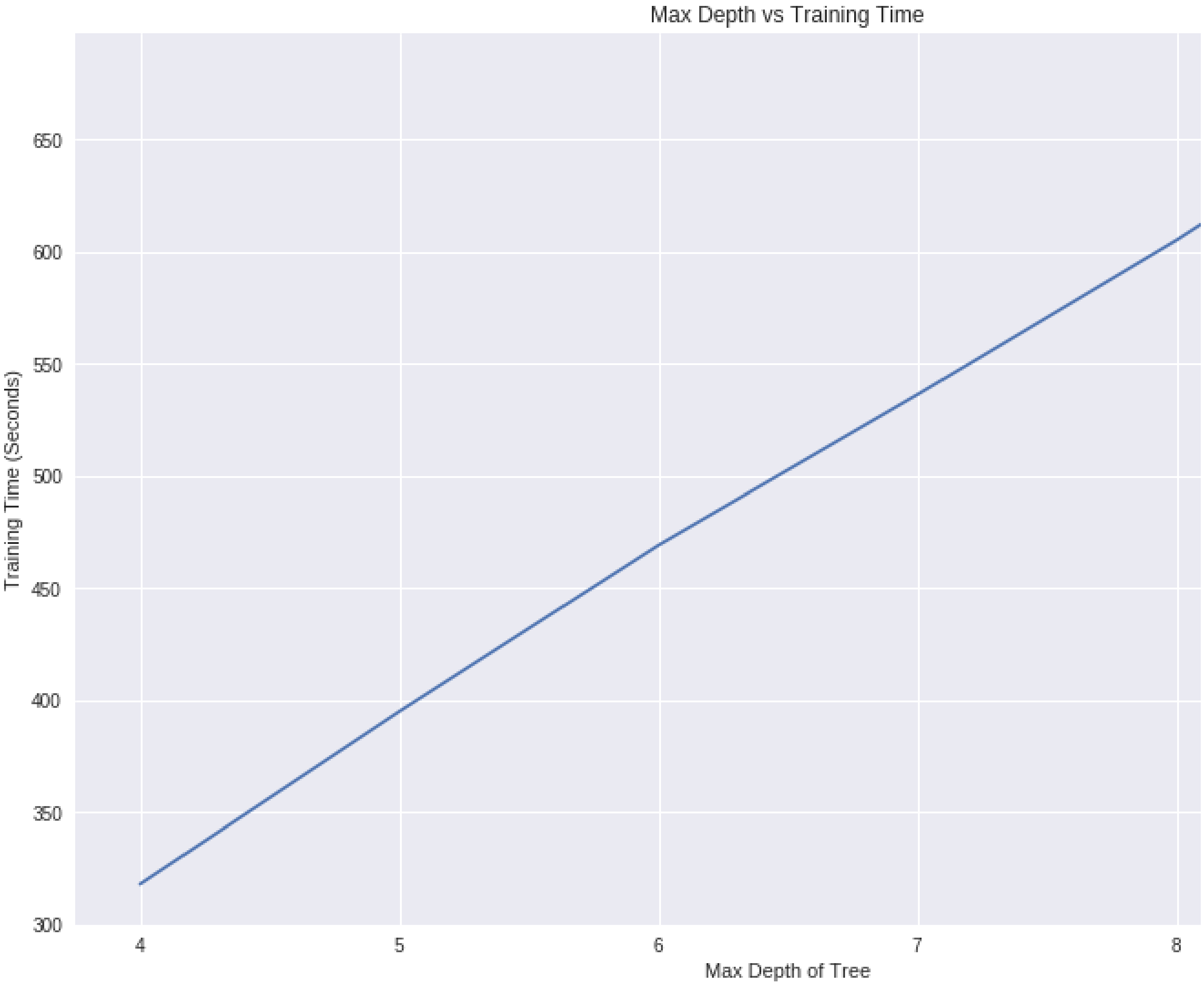
```
elapsed = []
results = []
# Nested Cross Validation :
from sklearn import cross_validation
for i in range(4,10):
  start = time.time()
  val_size = 0.1
  X_train, X_val, Y_train, Y_val = cross_validation.train_test_split(data, loss, test_siz
  model = XGBRegressor(seed=2016,max_depth=i)
  model.fit(X_train,Y_train,eval_metric='mae')
  end = time.time()
  elapsed.append(end-start)
  results.append(mean_absolute_error(np.expm1(Y_val),np.expm1(model.predict(X_val))))
  print "Performance:"+str(results[-1])
  print "Time:"+str(elapsed[-1])
```

```
/usr/local/lib/python2.7/dist-packages/sklearn/cross_validation.py:41: Deprecat
  "This module will be removed in 0.20.", DeprecationWarning)
Performance:1204.4408884114928
Time:317.990732908
Performance:1167.7376902300489
Time:394.815212011
Performance:1181.042126941717
Time:469.015637159
Performance:1149.8536614067507
Time:536.06773591
Performance:1164.6145181493175
Time:605.0114851
Performance:1151.1771897726023
Time:679.411887884
```

```
results = [1204.4408884114928,1167.7376902300489,1181.042126941717,1149.8536614067507,116
plt.figure(figsize=(14,9))
plt.plot(np.arange(4,10),results)
plt.title('MAE (CV) vs Max Depth of Decision Tree')
plt.ylabel('Mean Absolute Error')
plt.xlabel('Max Depth of Tree')
plt.show()
```



```
plt.figure(figsize=(14,9))
plt.plot(np.arange(4,10),elapsed)
plt.xlabel('Max Depth of Tree')
plt.ylabel('Training Time (Seconds)')
plt.title('Training Time vs Max Depth')
plt.show()
```



```
model = XGBRegressor(seed=2016,max_depth=7)
model.fit(data,loss,eval_metric='mae')
```

```
predictions = np.expm1(model.predict(dataTestE))
predLoss = pd.Series(predictions,name='loss')
submission  = pd.concat([df_test['id'],predLoss],axis=1)
```

```
submission.to_csv('submission-XGBOOST3-7Depth.csv',index=False)
files.download('submission-XGBOOST3-7Depth.csv')
```

```
from sklearn.model_selection import cross_val_score

model = XGBRegressor(seed=2016,max_depth=7)
scores = cross_val_score(model, data, loss, cv=4)
```

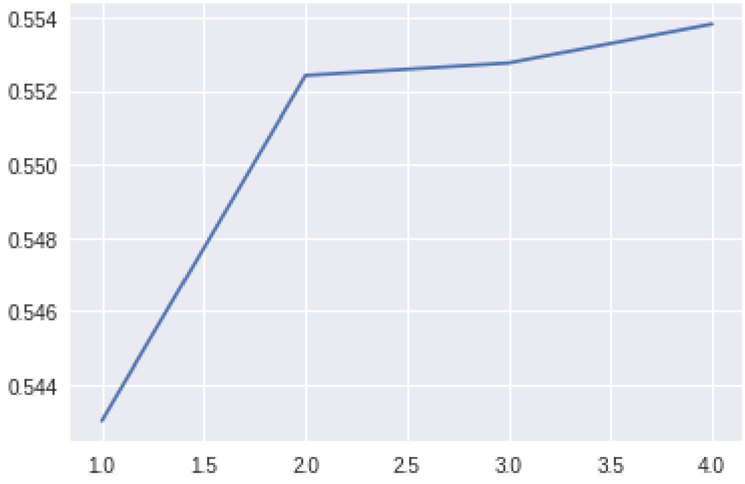
Cross Validated Accuracy:

```
scores
```

```
array([0.5430061 , 0.55242186, 0.55276236, 0.55382748])
```

```
plt.plot(np.arange(1,5),scores)
```

```
[<matplotlib.lines.Line2D at 0x7f1db375d690>]
```


0.554
0.552
0.550
0.548
0.546
0.544
1.0
1.5
2.0
2.5
3.0
3.5
4.0