Kaggle is a website that hosts ML competitions. In this lab, you will participate in a Kaggle competition with other students in this class! The top 3 people will earn up to bonus points. To join the competition, visit this link. You will need to register an account with Kaggle, but you can use your Google account.

In [23]:

```
# https://www.kaggle.com/c/beer2019/overview
# predict the bitterness of the beer

#import warnings
#warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np

df_train = pd.read_csv("dataset/beer_train.csv", index_col="id")
df_train.head(5)
```

Out[23]:

id	abv	available	description	glass	ibu	isOrganic	name	originalGravity	srm
0	8.2	Available at the same time of year, every year.	A Belgian-Abbey-Style Tripel that is big in al...	NaN	31.0	N	LoonyToonTripel	1.070	8
1	5.7	Available at the same time of year, every year.	Covert Hops is a crafty ale. Its stealthy dark...	Pint	45.0	N	Covert Hops	1.056	35
2	5.8	Available at the same time of year, every year.	This is a traditional German-style Marzen char...	Mug	25.0	N	Oktoberfest	1.048	10
3	5.5	Available year round as a staple beer.	A West Coast-Style Pale Ale balancing plenty o...	Pint	55.0	N	Pale Ale	1.044	5
4	4.8	Available year round as a staple beer.	This Bombshell has a tantalizing crisp and cle...	Pint	11.4	N	Head Turner Blonde Ale	1.045	3

In [24]:

```
# Later, we will ask you to read-in the beer_test.csv file into a dataframe.
# uncomment the following code, and proceed to apply the same transformations to df_test a
s we did for df_train

df_test = pd.read_csv("dataset/beer_test.csv", index_col="id")
df_test.head(5)
```

Out[24]:

id	abv	available	description	glass	ibu	isOrganic	name	originalGravity	srm
6000	10.0	Limited availability.	A classic Belgian Trappist style strong ale wi...	Tulip	NaN	N	She WILL!	1.084	17
6001	5.2	Available year round as a staple beer.	An American-style of Pale Ale brewed with a ba...	Pint	NaN	N	Defender American Pale Ale	1.044	22
6002	4.0	Available during the winter months.	This amber wheat ale has a balanced malt body,...	Tulip	NaN	Y	Hazel	1.036	19
6003	10.2	Available year round as a staple beer.	A uniquely large beer developed by taking our ...	Pint	NaN	N	Cinderella's Twin Double IPA	1.087	11
6004	6.0	Limited availability.	An American red ale with crisp hop flavor.	NaN	NaN	N	Independence Ale	1.048	14

In [25]:

```
df_train.shape
```

Out[25]:

```
(6000, 9)
```

In [26]:

```
#df_train.shape
df_train.describe()
```

Out[26]:

	abv	ibu	originalGravity
count	6000.000000	6000.000000	6000.000000
mean	6.502388	41.196878	1.053993
std	1.833054	28.846228	0.021747
min	0.200000	1.000000	1.000000
25%	5.200000	22.000000	1.044000
50%	6.000000	33.000000	1.050000
75%	7.400000	55.000000	1.060000
max	19.700000	1000.000000	1.980000

In [27]:

```
# print the nulls by column
df_train.isna().sum()
```

Out[27]:

```
abv                 0
available           0
description       479
glass            2342
ibu                 0
isOrganic           0
name                0
originalGravity     0
srm                 0
dtype: int64
```

In [28]:

```
df_test.isna().sum()
```

Out[28]:

```
abv                  0
available            0
description        397
glass             1919
ibu               4753
isOrganic            0
name                 0
originalGravity      0
srm                  0
dtype: int64
```

In [29]:

```
# drop the column 'glass' since it has too many NULLs
# *** TODO ***
df_train.drop('glass', axis = 1, inplace = True)
df_test.drop('glass', axis = 1, inplace = True)
```

In [30]:

```
# isOrganic is a binary value, lets convert it to 0/1
# *** TODO ***
df_train['isOrganic'] = df_train['isOrganic'].map({'N':0, 'Y':1})
df_test['isOrganic'] = df_test['isOrganic'].map({'N':0, 'Y':1})
```

In [31]:

```
# Lets examine the column 'available'. Note that it is a categorical variable
df_train["available"].value_counts()
```

Out[31]:

```
Available year round as a staple beer.             2812
Limited availability.                              1204
Available at the same time of year, every year.    1192
Beer is not available.                              255
Available during the summer months.                 159
Available during the winter months.                 142
Available during the fall months.                   141
Available during the spring months.                  95
Name: available, dtype: int64
```

In [32]:

```
df_test["available"].value_counts()
```

Out[32]:

```
Available year round as a staple beer.             2231
Available at the same time of year, every year.     959
Limited availability.                               945
Beer is not available.                              246
Available during the winter months.                 106
Available during the summer months.                 101
Available during the fall months.                    92
Available during the spring months.                  73
Name: available, dtype: int64
```

In [33]:

```
df_train['ibu'].describe()
```

Out[33]:

```
count    6000.000000
mean       41.196878
std        28.846228
min         1.000000
25%        22.000000
50%        33.000000
75%        55.000000
max      1000.000000
Name: ibu, dtype: float64
```

In [34]:

```
#indexes = df_train[df_train['ibu'] > 120].index

#df_train.drop(indexes , inplace = True)
```

In [35]:

```
df_train.dtypes
```

Out[35]:

```
abv               float64
available          object
description        object
ibu               float64
isOrganic           int64
name               object
originalGravity   float64
srm                object
dtype: object
```

In [36]:

```
df_train['ibu'].describe()
```

Out[36]:

```
count    6000.000000
mean       41.196878
std        28.846228
min         1.000000
25%        22.000000
50%        33.000000
75%        55.000000
max      1000.000000
Name: ibu, dtype: float64
```

In [37]:

```
# Convert column 'available' using oneHotEncoding/get_dummies

df_train = pd.concat([df_train, pd.get_dummies(df_train['available'], prefix='available',
dummy_na=True)],
                     axis=1).drop(['available'], axis=1)

df_test = pd.concat([df_test, pd.get_dummies(df_test['available'], prefix='available', dum
my_na=True)],
                     axis=1).drop(['available'], axis=1)


# It probably makes sense to rename those columns to something shorter.
# Look up how to rename columns (inplace) in a dataframe

df_train.columns = ['abv', 'description', 'ibu', 'isOrganic', 'name', 'originalGravity',
'srm', 'av_same_time',
                    'av_fall', 'av_spring', 'av_summer', 'av_winter', 'av_staple', 'av_not'
, 'av_limited', 'av_nan']
df_test.columns = ['abv', 'description', 'ibu', 'isOrganic', 'name', 'originalGravity', 's
rm', 'av_same_time',
                    'av_fall', 'av_spring', 'av_summer', 'av_winter', 'av_staple', 'av_not'
, 'av_limited', 'av_nan']
```

In [38]:

```
# You might have noticed that SRM appears to be numeric, but its not listed as a numeric c
olumn.
# There is probably some rows that are strings that need to be fixed.
# Correct rows in column 'SRM' so that string values 'Over 40' are replaced with 40
df_train['srm'] = pd.to_numeric(df_train['srm'].replace('Over 40', 40))
df_test['srm'] = pd.to_numeric(df_test['srm'].replace('Over 40', 40))
```

In [39]:

```
df_train.dtypes
```

Out[39]:

```
abv                float64
description         object
ibu                float64
isOrganic            int64
name                object
originalGravity    float64
srm                  int64
av_same_time         uint8
av_fall              uint8
av_spring            uint8
av_summer            uint8
av_winter            uint8
av_staple            uint8
av_not               uint8
av_limited           uint8
av_nan               uint8
dtype: object
```

Setting up model

In [40]:

```
X_train = df_train.copy(deep=True) # copy the dataframe
X_train.drop(['description', 'name', 'ibu'], axis = 1, inplace = True) # use only 1 featur
e for trainings to start off
X_train.head(2)
```

Out[40]:

id	abv	isOrganic	originalGravity	srm	av_same_time	av_fall	av_spring	av_summer	av_winte
0	8.2	0	1.070	8	1	0	0	0	
1	5.7	0	1.056	35	1	0	0	0	

In [41]:

```
X_test = df_test.copy(deep=True) # copy the dataframe
X_test.drop(['description', 'name', 'ibu'], axis = 1, inplace = True) # use only 1 feature
for trainings to start off
X_test.head(2)
```

Out[41]:

id	abv	isOrganic	originalGravity	srm	av_same_time	av_fall	av_spring	av_summer	av_w
6000	10.0	0	1.084	17	0	0	0	0	
6001	5.2	0	1.044	22	0	0	0	0	

In [42]:

```
from sklearn.linear_model import LinearRegression

y_train = df_train["ibu"]

model = LinearRegression()
model.fit(X=X_train, y=y_train)
Y_predict = model.predict(X_test) # TODO , change X=X_train to X_test once you create X_te
st per the instructions below
```

In [43]:

```
# Examine the model coefficients, what can you conclude ??
model.coef_

# ** TODO ** Observations / comments
```

Out[43]:

```
array([ 7.24200055e+00, -7.43751084e-02,  1.29501477e+02, -1.71948874e-01,
        6.61557890e-01, -7.53252589e+00,  8.25975631e-01, -1.92186109e+00,
       -2.49812507e+00,  5.70090609e+00,  2.08202353e+00,  2.68204891e+00,
        0.00000000e+00])
```

In [44]:

```
output = pd.DataFrame(data = {"id":df_test.index, "ibu":Y_predict}) # first need to create
df_test

output.to_csv(path_or_buf = "results.csv", index = False, quoting=3, sep=',')
```

Next, we want to evaluate this model on the test dataset and submit our predictions to Kaggle.

1) Evaluate the code from the first cell and add a new dataframe called df_test to read in the beer_test.csv file.

2) For each step / transformation that was applied to the training dataset, apply it to the test dataset

3) Go to Kaggle, create an account

4) Submit the results.csv file for this compeition to Kaggle

5) What score did you get?? Score: 25.9

6) Now, experiment with a differnt set of features. What is the best score recieved? What subset of features / columns did you use?

Build model using validation dataset

Now, lets use cross-validation to train our model.

In [47]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline

# pipeline and fit to training set
pipeline = make_pipeline(
    StandardScaler(),
    SGDRegressor()
)
pipeline.fit(X=X_train, y=y_train)

scores = cross_val_score(pipeline,
                         X=X_train,
                         y=y_train,
                         cv=5)  # cv refers to number of cross-validation cuts
scores
```

Out[47]:

```
array([0.2637006 , 0.28076459, 0.09181001, 0.2904624 , 0.1695955 ])
```

In [48]:

```
Y_predict = pipeline.predict(X = X_test)
output = pd.DataFrame(data = {"id":df_test.index, "ibu":Y_predict})

output.to_csv(path_or_buf = "results.csv", index = False, quoting = 3, sep = ',')
```

Resubmit your results.csv file to Kaggle. Did cross-validation improve accuracy this time around??
Nope, I did 10 cross validation and my resubmission score is 26.12.
Nope, I tried 5 cross validation and my resubmission score is 26.017.

Text Features

Now, lets examine how we can use text columns as features. We will transform the column 'description' by

1) finding the relevant features (keywords) in the description and assigning them weights based on 'importance'

2) Creating a dataframe for each feature (or keyword) with the given weight

3) Combine the new description features with the original df_train dataframe

In [49]:

```
df_train['description'].fillna('', inplace=True) # fill Nulls with empty space
df_test['description'].fillna('', inplace=True)
```

In [50]:

```
#df_train.isna().sum() # list number of Nulls
```

In [51]:

```
# Use TFIDF to find important features from the 'description' column
from sklearn.feature_extraction.text import TfidfVectorizer

# settings that you use for count vectorizer will go here
tfidf_vectorizer_train = TfidfVectorizer(lowercase=True,use_idf=True,
                                         max_features=100, stop_words='english')

tfidf_vectorizer_test = TfidfVectorizer(lowercase=True,use_idf=True,
                                         max_features=100, stop_words='english')


# just send in all your docs here
tfidf_vectorizer_vectors_train = tfidf_vectorizer_train.fit_transform(df_train["descriptio
n"])
tfidf_vectorizer_vectors_test = tfidf_vectorizer_test.fit_transform(df_test["description"
])


df_train_tfidf = pd.DataFrame(tfidf_vectorizer_vectors_train.todense(),
                              columns=tfidf_vectorizer_train.get_feature_names())

df_test_tfidf = pd.DataFrame(tfidf_vectorizer_vectors_test.todense(),
                              columns=tfidf_vectorizer_test.get_feature_names())
```

In [52]:

```
df_train = pd.concat([df_train, df_train_tfidf], axis=1)
```

In [53]:

```
df_test_tfidf.index = range(6000,len(df_test_tfidf) + 6000)
```

In [54]:

```
df_test = pd.concat([df_test, df_test_tfidf], axis=1)
```

In [55]:

```
X_train = df_train.copy(deep = True) # copy the dataframe
X_train.drop(['description', 'name', 'ibu'], axis = 1, inplace = True)
X_train.head(2)
```

Out[55]:

	abv	isOrganic	originalGravity	srm	av_same_time	av_fall	av_spring	av_summer	av_winter
0	8.2	0	1.070	8	1	0	0	0	0
1	5.7	0	1.056	35	1	0	0	0	0

2 rows × 113 columns

In [56]:

```
y_train = df_train["ibu"]
```

In [57]:

```
X_test = df_test.copy(deep = True) # copy the dataframe
X_test.drop(['description', 'name', 'ibu'], axis = 1, inplace = True)
X_test.head(2)
```

Out[57]:

	abv	isOrganic	originalGravity	srm	av_same_time	av_fall	av_spring	av_summer	av_w
6000	10.0	0	1.084	17	0	0	0	0	
6001	5.2	0	1.044	22	0	0	0	0	

2 rows × 113 columns

In [58]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline

# pipeline and fit to training set
pipeline = make_pipeline(
    StandardScaler(),
    SGDRegressor()
)
pipeline.fit(X=X_train, y=y_train)

scores = cross_val_score(pipeline,
                         X=X_train,
                         y=y_train,
                         cv=5)  # cv refers to number of cross-validation cuts
scores
```

Out[58]:

```
array([0.52720865, 0.56546462, 0.36592962, 0.51347067, 0.27949936])
```

In [59]:

```
Y_predict = pipeline.predict(X=X_test)
output=pd.DataFrame(data={"id":df_test.index,"ibu":Y_predict})

output.to_csv(path_or_buf="results.csv",index=False,quoting=3,sep=',')
```

Rerun the model with the additional features. Did you receieve a better score??
Score: 24.97

Additional Models

Try additional models or approaches to try to get a better score. Submit your completed notebook and a snapshot of the best score you obtained on Kaggle to iLearn.

In [60]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor

regr_1 = DecisionTreeRegressor(max_depth=4)
regr_2 = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4), n_estimators=300)
regr_1.fit(X_train, y_train)
regr_2.fit(X_train, y_train)
```

Out[60]:

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(ccp_alpha=0.0,
                                                       criterion='mse',
                                                       max_depth=4,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=
0.0,
                                                       min_impurity_split=Non
e,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_le
af=0.0,
                                                       presort='deprecated',
                                                       random_state=None,
                                                       splitter='best'),
                  learning_rate=1.0, loss='linear', n_estimators=300,
                  random_state=None)
```

In [61]:

```
Y_predict = regr_2.predict(X=X_test)
output=pd.DataFrame(data={"id":df_test.index,"ibu":Y_predict})

output.to_csv(path_or_buf="results.csv",index=False,quoting=3,sep=',')
```

Score: 42.7