Exam2

March 7, 2019

1 Exam 2

In this exam, you will request data from the Zomato API and then train machine learning models on the data that you gather. You should have already registered an API key with Zomato, which will entitle you to 1000 API requests per day. This exam can be completed with fewer than 200 API requests.

```
In [1]: %matplotlib inline

        import numpy as np
        import pandas as pd
        import requests

        apikey = "49f040db06fd803e2a8465d4a241daa3"
```

2 Question 1 (4 points)

Determine the Zomato city ID for Chicago, IL. Then, use this city ID to create a `DataFrame` containing all of the cuisines in Chicago, along with their IDs. Display this `DataFrame`.

```
In [2]: url = ("https://developers.zomato.com/api/v2.1/"
               "location_details?entity_id=292&entity_type=city")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago = resp.json()
```

```
In [3]: import json
        from pandas.io.json import json_normalize

        chicago_df = json_normalize(chicago["best_rated_restaurant"])
        chicago_df.head()
```

```
Out[3]:    restaurant.R.res_id                 restaurant.apikey  \
        0             16736014  49f040db06fd803e2a8465d4a241daa3
        1             16737455  49f040db06fd803e2a8465d4a241daa3
        2             16752484  49f040db06fd803e2a8465d4a241daa3
        3             16734364  49f040db06fd803e2a8465d4a241daa3
```

```
4            16753200  49f040db06fd803e2a8465d4a241daa3

   restaurant.average_cost_for_two restaurant.book_again_url  \
0                               40
1                               30
2                               35
3                               90
4                               85

  restaurant.book_form_web_view_url                restaurant.cuisines  \
0                                                                Pizza
1                                   American, Italian, Burger, Sandwich
2                                                              Mexican
3                                                      Brazilian, Steak
4                                                          New American

  restaurant.currency           restaurant.deeplink  \
0                   $  zomato://restaurant/16736014
1                   $  zomato://restaurant/16737455
2                   $  zomato://restaurant/16752484
3                   $  zomato://restaurant/16734364
4                   $  zomato://restaurant/16753200

                              restaurant.events_url  \
0  https://www.zomato.com/chicago/lou-malnatis-pi...
1  https://www.zomato.com/chicago/portillos-hot-d...
2  https://www.zomato.com/chicago/xoco-river-nort...
3  https://www.zomato.com/chicago/fogo-de-chao-br...
4  https://www.zomato.com/chicago/girl-the-goat-w...

                          restaurant.featured_image  \
0  https://b.zmtcdn.com/data/res_imagery/16736012...
1  https://b.zmtcdn.com/data/res_imagery/16737455...
2  https://b.zmtcdn.com/data/res_imagery/16752484...
3  https://b.zmtcdn.com/data/res_imagery/16734364...
4  https://b.zmtcdn.com/data/res_imagery/16753200...

             ...               \
0            ...
1            ...
2            ...
3            ...
4            ...

                              restaurant.photos_url  restaurant.price_range  \
0  https://www.zomato.com/chicago/lou-malnatis-pi...                       2
1  https://www.zomato.com/chicago/portillos-hot-d...                       2
2  https://www.zomato.com/chicago/xoco-river-nort...                       2
```

```
3  https://www.zomato.com/chicago/fogo-de-chao-br...                        4
4  https://www.zomato.com/chicago/girl-the-goat-w...                        4

   restaurant.switch_to_order_menu  \
0                                0
1                                0
2                                0
3                                0
4                                0

                                   restaurant.thumb  \
0  https://b.zmtcdn.com/data/res_imagery/16736012...
1  https://b.zmtcdn.com/data/res_imagery/16737455...
2  https://b.zmtcdn.com/data/res_imagery/16752484...
3  https://b.zmtcdn.com/data/res_imagery/16734364...
4  https://b.zmtcdn.com/data/res_imagery/16753200...

                                     restaurant.url  \
0  https://www.zomato.com/chicago/lou-malnatis-pi...
1  https://www.zomato.com/chicago/portillos-hot-d...
2  https://www.zomato.com/chicago/xoco-river-nort...
3  https://www.zomato.com/chicago/fogo-de-chao-br...
4  https://www.zomato.com/chicago/girl-the-goat-w...

   restaurant.user_rating.aggregate_rating  \
0                                      4.7
1                                      4.9
2                                      4.9
3                                      4.6
4                                      4.8

   restaurant.user_rating.has_fake_reviews  \
0                                        0
1                                        0
2                                        0
3                                        0
4                                        0

   restaurant.user_rating.rating_color  restaurant.user_rating.rating_text  \
0                               3F7E00                           Excellent
1                               3F7E00                           Excellent
2                               3F7E00                           Excellent
3                               3F7E00                           Excellent
4                               3F7E00                           Excellent

  restaurant.user_rating.votes
0                          789
1                         1023
```

```
2                              737
3                              601
4                              743

[5 rows x 43 columns]
```

3 Question 2 (4 points)

Get the top 40 restaurants (sorted in `desc` order by `rating`) in Chicago for the following cuisines:

- Chinese
- Italian
- Mexican

Store the 120 results in a single `DataFrame` with the following columns:

- `res_id`: the ID for the Zomato restaurant
- `name`
- all of the `location` features (i.e., `address`, `locality`, `latitude`, `longitude`, `zipcode`, etc.)
- all of the `user_rating` features (i.e., `aggregate_rating`, `votes`, etc.)

Display this `DataFrame`.

```
In [4]: url = ("https://developers.zomato.com/api/v2.1/search?entity_id=292&entity"
               "_type=city&start=0&count=40&cuisines=25&sort=rating&order=desc"
        )
        resp = requests.get(url, headers={"user-key": apikey})
        chicago_chinese_part1 = resp.json()

        chicago_chinese_df_part1 = json_normalize(chicago_chinese_part1["restaurants"])
```

```
In [5]: url = ("https://developers.zomato.com/api/v2.1/search?"
            "entity_id=292&entity_type=city&start=20&cuisines=25&sort=rating&order=desc")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago_chinese_part2 = resp.json()

        chicago_chinese_df_part2 = json_normalize(chicago_chinese_part2["restaurants"])
```

```
In [6]: url = ("https://developers.zomato.com/api/v2.1/search?"
          "entity_id=292&entity_type=city&start=0&cuisines=55&sort=rating&order=desc")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago_italian_part1 = resp.json()

        chicago_italian_part1_df = json_normalize(chicago_italian_part1["restaurants"])
```

```
In [7]: url = ("https://developers.zomato.com/api/v2.1/search?entity_id"
             "=292&entity_type=city&start=20&cuisines=55&sort=rating&order=desc")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago_italian_part2 = resp.json()

        chicago_italian_part2_df = json_normalize(chicago_italian_part2["restaurants"])
```

```
In [8]: url = ("https://developers.zomato.com/api/v2.1/search?entity_id="
            "292&entity_type=city&start=0&cuisines=73&sort=rating&order=desc")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago_mexican_part1 = resp.json()

        chicago_mexican_part1_df = json_normalize(chicago_mexican_part1["restaurants"])
```

```
In [9]: url = ("https://developers.zomato.com/api/v2.1/search?entity_id="
            "292&entity_type=city&start=20&cuisines=73&sort=rating&order=desc")

        resp = requests.get(url, headers={"user-key": apikey})
        chicago_mexican_part2 = resp.json()

        chicago_mexican_part2_df = json_normalize(chicago_mexican_part2["restaurants"])
```

```
In [10]: chicago_chinese_df_part1["my_cuisine"] = "Chinese"
         chicago_chinese_df_part2["my_cuisine"] = "Chinese"
         chicago_italian_part1_df["my_cuisine"] = "Italian"
         chicago_italian_part2_df["my_cuisine"] = "Italian"
         chicago_mexican_part1_df["my_cuisine"] = "Mexican"
         chicago_mexican_part2_df["my_cuisine"] = "Mexican"

         all_dfs = [chicago_chinese_df_part1, chicago_chinese_df_part2,
                    chicago_italian_part1_df, chicago_italian_part2_df,
                    chicago_mexican_part1_df, chicago_mexican_part2_df]

         chicago_restaurants = pd.concat(all_dfs)
```

```
In [11]: my_index = range(0,120)

         chicago_restaurants["my_index"] = my_index

         chicago_restaurants = chicago_restaurants.set_index(chicago_restaurants["my_index"])
```

```
In [12]: final_chicago_df = chicago_restaurants[["restaurant.R.res_id", "my_cuisine",
                             "restaurant.location.address", "restaurant.location.city",
                             "restaurant.location.city_id", "restaurant.location.country_id",
                             "restaurant.location.latitude", "restaurant.location.locality",
                             "restaurant.location.locality_verbose",
                             "restaurant.location.longitude", "restaurant.location.zipcode",
```

```
                        "restaurant.name", "restaurant.user_rating.aggregate_rating",
                        "restaurant.user_rating.rating_color",
                        "restaurant.user_rating.rating_text",
                        "restaurant.user_rating.votes"]]

final_chicago_df.head()
```

Out[12]:

```
          restaurant.R.res_id my_cuisine  \
my_index
0                    16735363    Chinese
1                    16749821    Chinese
2                    16743712    Chinese
3                    16747211    Chinese
4                    16751533    Chinese

                              restaurant.location.address  \
my_index
0                                 521 Davis Street 60201
1                                  200 E Golf Road 60173
2             9560 S. Kedzie Avenue, Evergreen Park 60805
3         1003 W. Ogden Avenue, Suite B, Naperville 60563
4                     537 Green Bay Road, Wilmette 60091

         restaurant.location.city  restaurant.location.city_id  \
my_index
0                         Chicago                          292
1                         Chicago                          292
2                         Chicago                          292
3                         Chicago                          292
4                         Chicago                          292

          restaurant.location.country_id restaurant.location.latitude  \
my_index
0                                    216                42.0460805556
1                                    216                42.0507000000
2                                    216                41.7197000000
3                                    216                41.7848777778
4                                    216                42.0748055556

         restaurant.location.locality restaurant.location.locality_verbose  \
my_index
0                            Evanston                   Evanston, Chicago
1                          Schaumburg                 Schaumburg, Chicago
2                      Evergreen Park             Evergreen Park, Chicago
3                          Naperville                 Naperville, Chicago
4                            Wilmette                   Wilmette, Chicago

         restaurant.location.longitude restaurant.location.zipcode  \
```

```
my_index
0                      -87.6790305556                      60201
1                      -88.0742638889                      60173
2                      -87.7020000000                      60805
3                      -88.1680916667                      60563
4                      -87.7077638889                      60091

                    restaurant.name restaurant.user_rating.aggregate_rating  \
my_index
0                  Joy Yee's Noodles                                    4.5
1            Yu's Mandarin Restaurant                                   4.5
2               Chi Tung Restaurant                                     4.5
3                   Chinese Kitchen                                     4.4
4          Tsing Tao Mandarin Chinese                                   4.4

         restaurant.user_rating.rating_color  \
my_index
0                                     3F7E00
1                                     3F7E00
2                                     3F7E00
3                                     5BA829
4                                     5BA829

         restaurant.user_rating.rating_text restaurant.user_rating.votes
my_index
0                                 Excellent                          161
1                                 Excellent                          158
2                                 Excellent                          115
3                                 Very Good                           31
4                                 Very Good                           22
```

4 Question 3 (4 points)

For each of the 120 restaurants that you identified in Question 2, get the 5 most recent reviews. Store all of the reviews in a `DataFrame` with the following columns:

- `rating`
- `review_text`
- all of the user features (such as `name`, `foodie_level`, `foodie_level_num`)

Display this `DataFrame`.

In [13]:
```
import time

all_reviews_df = pd.DataFrame()

for res_id in final_chicago_df["restaurant.R.res_id"]:
    url = "https://developers.zomato.com/api/v2.1/reviews?res_id={}".format(res_id)
```

```
            resp = requests.get(url, headers={"user-key": apikey})
            reviews = resp.json()
            reviews_df = json_normalize(reviews["user_reviews"])
            all_reviews_df = all_reviews_df.append(reviews_df)

            time.sleep(.5)
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py:6211: FutureWarning: Sorting becau
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  sort=sort)
```

In [14]:
```
my_index = range(0,578)
all_reviews_df["my_index"] = my_index

all_reviews_df = all_reviews_df.set_index(all_reviews_df["my_index"])
```

In [15]:
```
all_reviews_df.head()
```

Out[15]:
```
          review.comments_count  review.id  review.likes  review.rating  \
my_index
0                             0   32160086             0            5.0
1                             0   26207391             1            5.0
2                             0   24891125             0            4.0
3                             0   24622388             1            3.5
4                             0   16100548             0            0.0

         review.rating_color review.rating_text  \
my_index
0                     305D02            Insane!
1                     305D02            Insane!
2                     5BA829             Great!
3                     9ACD32        Good Enough
4                     CBCBC8          Not rated

                                         review.review_text  \
my_index
0         Lively ambiance, but its all about the food. T...
1         It was very good food a customer service  Has ...
2         Yummy yummy yummy! This place is a gold mine. ...
3         Huge portions. good food, and they actually ma...
4         Stay away from this rapidly declining restaura...

         review.review_time_friendly  review.timestamp  \
```

```
my_index
0                      Oct 23, 2017        1508781075
1                      Feb 07, 2016        1454800032
2                      Sep 11, 2015        1441916515
3                      Aug 15, 2015        1439587312
4                      Nov 13, 2013        1384366238

         review.user.foodie_color review.user.foodie_level  \
my_index
0                          ffd35d                   Foodie
1                          ffd35d                   Foodie
2                          ffae4f               Big Foodie
3                          f58552             Super Foodie
4                          ffd35d                   Foodie

          review.user.foodie_level_num review.user.name  \
my_index
0                                    3       Jason Jobe
1                                    2    Tarek Anthony
2                                    6          Jalpa T
3                                    8           Prapti
4                                    1          Mtnsguy

         review.user.profile_deeplink  \
my_index
0                  zomato://u/30860504
1                  zomato://u/33462700
2                  zomato://u/30483347
3                   zomato://u/1510791
4                  zomato://u/23884812

                                  review.user.profile_image  \
my_index
0          https://b.zmtcdn.com/data/user_profile_picture...
1          https://b.zmtcdn.com/data/user_profile_picture...
2          https://b.zmtcdn.com/data/user_profile_picture...
3          https://b.zmtcdn.com/data/user_profile_picture...
4          https://b.zmtcdn.com/images/user_avatars/mug_2...

                                    review.user.profile_url  \
my_index
0          https://www.zomato.com/users/jason-jobe-308605...
1          https://www.zomato.com/users/tarek-anthony-334...
2          https://www.zomato.com/samosapop?utm_source=ap...
3          https://www.zomato.com/Praptisahni?utm_source=...
4          https://www.zomato.com/users/mtnsguy-23884812?...

         review.user.zomato_handle  my_index
```

```
my_index
0                              NaN         0
1                              NaN         1
2                        samosapop         2
3                      Praptisahni         3
4                              NaN         4
```

5 Question 4 (6 points)

Let's use the restaurants data that you obtained in Question 2 to train a machine learning model to predict whether a restaurant serves Chinese, Italian, or Mexican cuisine, given just the `latitude` and the `longitude` of the restaurant. (For a 1 point penalty, you can download `restaurants.csv` from PolyLearn, upload it to the current directory, and read in the file.)

Train a k-nearest neighbors model. Determine the optimal value of k. Calculate an estimate of the test precision and recall of your final model. Interpret these values in the context of this application.

In [16]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

In [17]:
```
import warnings
warnings.filterwarnings('ignore')

features = ["restaurant.location.latitude", "restaurant.location.longitude"]

X_train = final_chicago_df[features].astype('float64')
y_train = final_chicago_df["my_cuisine"]

is_Chinese_train = (y_train == "Chinese")

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)

best_k = []
best_cv = []
best_accuracy = []
best_recall = []
best_precision = []
best_score = 0

for k in list(range(1,31)):
    model = KNeighborsClassifier(n_neighbors=k)
```

```
            model.fit(X_train_sc, y_train)

            pipeline = Pipeline([
                ("scaler", scaler),
                ("model", model)
            ])

            for cross in list(range(2,11)):
                f1 = cross_val_score(pipeline, X_train, is_Chinese_train,
                                     cv=cross, scoring="f1").mean()

                accuracy = cross_val_score(pipeline, X_train, y_train,
                                           cv=cross, scoring="accuracy").mean()

                recall = cross_val_score(pipeline, X_train, is_Chinese_train,
                                         cv=cross, scoring="recall").mean()

                precision = cross_val_score(pipeline, X_train, is_Chinese_train,
                                            cv=cross, scoring="precision").mean()

                if f1 > best_score:
                    best_score = f1
                    best_k.append(k)
                    best_cv.append(cross)
                    best_accuracy.append(accuracy)
                    best_recall.append(recall)
                    best_precision.append(precision)


        print("Neighbors: ", best_k.pop())
        print("Cross: ", best_cv.pop())
        print("F1: ", best_score)
        print("Accuracy: ", best_accuracy.pop())
        print("Recall: ", best_recall.pop())
        print("Precision: ", best_precision.pop())
```

```
Neighbors:  3
Cross:  2
F1:  0.494444444444
Accuracy:  0.425
Recall:  0.475
Precision:  0.51875
```

I have found the optimal value of k to be 3 neighbors. I trained models ranging from 1 neighbor all the way up to 30 neighbors. For each of these models, I tested cross validation folds ranging from 2 to 10. In order to reach this conclusion, I used the F1 score to compare which model was better. If the current model was better than any previous model, I saved the current model as the "best" and continued to compare the following models based on the F1 score.

6 Question 5 (6 points)

Let's use the reviews data that you obtained in Question 3 to train a machine learning model to predict a rating, given just the `review_text`. (For a 1 point penalty, you can download `reviews.csv` from PolyLearn, upload it to the current directory, and read in the file.)

You will have to first convert the text of the review into quantitative features. Instead of including every word that appears, it is usually better to restrict to words that appear at least m times, where m is a hyperparameter. Plot the training and the test RMSE of a 10-nearest neighbors model as a function of this hyperparameter m. What value of m is optimal? What is the test RMSE of this optimal model? Interpret the test RMSE in the context of this application.

Hint: The hyperparameter m corresponds to the `min_df=` argument of `CountVectorizer` and `TfidfVectorizer` in *scikit-learn*.

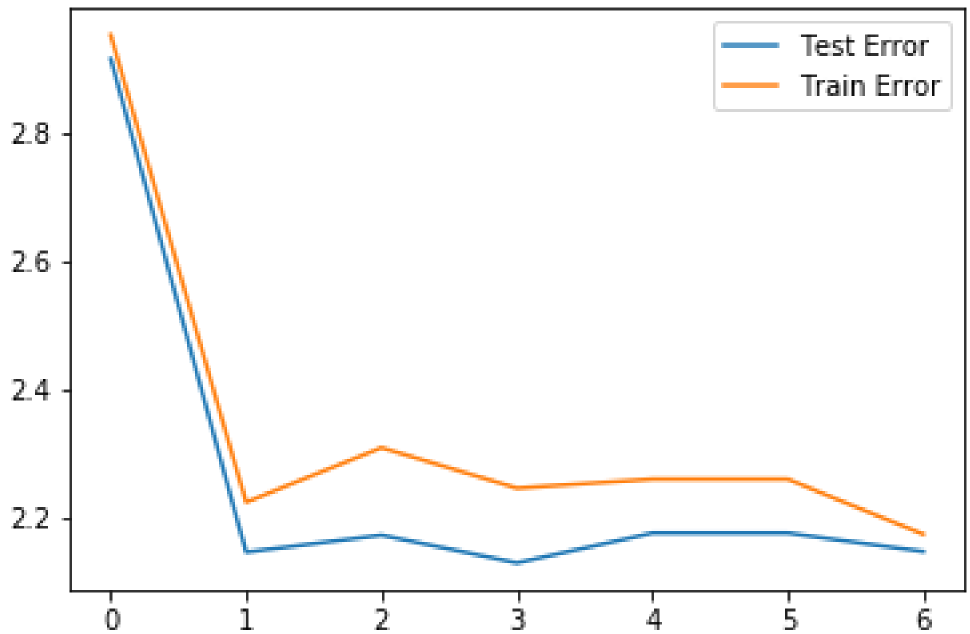
```
In [18]: all_reviews_df["review.review_text"] =
             all_reviews_df["review.review_text"].fillna("None")
```

```
In [19]: from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.neighbors import KNeighborsRegressor

         model1 = KNeighborsRegressor(n_neighbors=10)
         def test_error(m):
             m = m/10
             vec = TfidfVectorizer(norm=None, min_df=m) #, min_df=.5)
             vec.fit(all_reviews_df["review.review_text"])
             tf_idf_sparse = vec.transform(all_reviews_df["review.review_text"])
             tfidf = pd.DataFrame(tf_idf_sparse.toarray())

             X_train = tfidf
             y_train = all_reviews_df["review.rating"]

             scaler = StandardScaler()
             scaler.fit(X_train)
             X_train_sc = scaler.transform(X_train)


             model1 = KNeighborsRegressor(n_neighbors=10)
             model1.fit(X_train_sc, y_train)

             pipeline = Pipeline([
                 ("scaler", scaler),
                 ("model", model1)
             ])

             rmse = cross_val_score(pipeline, X_train, y_train,
                                    cv=5, scoring="neg_mean_squared_error").mean()

             return np.sqrt(np.mean(-rmse))
```

```python
model2 = KNeighborsRegressor(n_neighbors=10)
def train_error(m):
    m = m/10
    vec = TfidfVectorizer(norm=None, min_df=m) #, min_df=.5)
    vec.fit(all_reviews_df["review.review_text"])
    tf_idf_sparse = vec.transform(all_reviews_df["review.review_text"])
    tfidf = pd.DataFrame(tf_idf_sparse.toarray())

    X_train = tfidf
    y_train = all_reviews_df["review.rating"]

    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train_sc = scaler.transform(X_train)


    model2 = KNeighborsRegressor(n_neighbors=10)
    model2.fit(X_train_sc, y_train)

    pipeline = Pipeline([
        ("scaler", scaler),
        ("model", model2)
    ])

    rmse = cross_val_score(pipeline, X_train, y_train,
                           scoring="neg_mean_squared_error").mean()

    return np.sqrt(np.mean(-rmse))



ms = pd.Series(range(0, 7, 1))
ms.index = range(0, 7, 1)
test_error = ms.apply(test_error)
train_error = ms.apply(train_error)
```

In [20]:
```python
test_error.plot.line(label="Test Error", legend=True)
train_error.plot.line(label="Train Error", legend=True)
test_error.sort_values()
```

Out[20]:
```
3    2.130146
1    2.146974
6    2.147989
2    2.173584
4    2.176793
5    2.176793
0    2.915951
dtype: float64
```



I have the optimal value of "m" to be 3 which is technically .3 as a parameter in the min_df argument. The rmse from using 3 as my m is 2.130146. RMSE is the average standard deviation of each data point is from its predicted point, so in context, the rmse is measuring the average distance between my model's prediction for the review rating that each customer gave and the review rating the customer actually gave in the review.

I obtained this answer by creating 2 funcitons that essentially do the same thing - train_error does not uses cross_validation folds, while test_error uses 5 folds (chosen arbitrarily). I first used TfidfVectorizer to turn the reviews into numerical data to work with for each review. From here, it was setting the training data, scaling, making and fitting the model using pipeline from scikit learn.

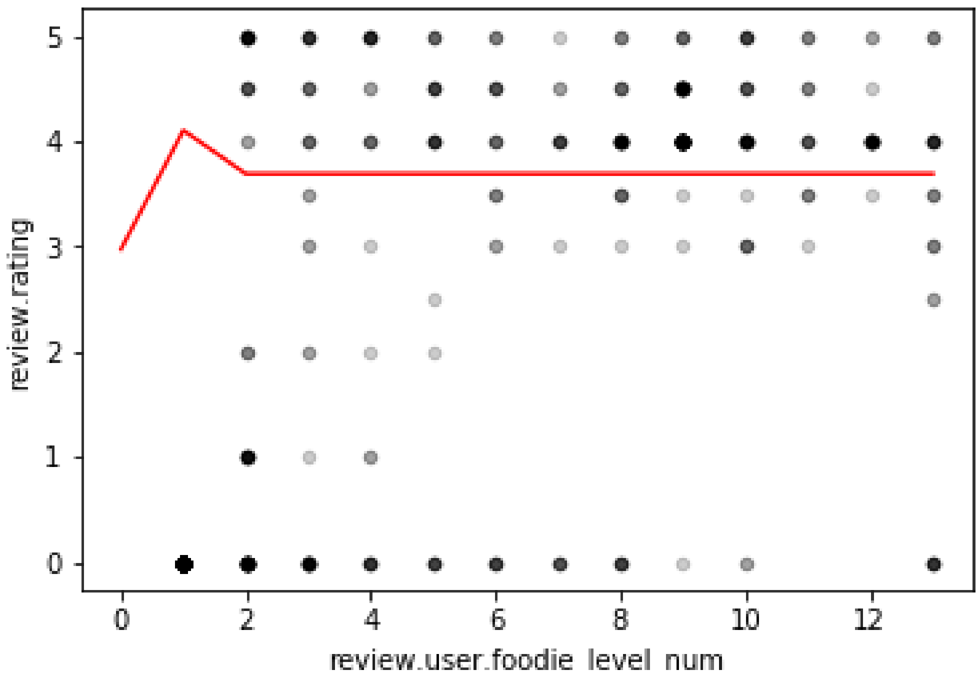
7 Question 6 (6 points)

Let's use the reviews data to train a machine learning model to predict the `rating`, given just the `foodie_level_num` of the user. Fit an 80-nearest neighbors model to predict `rating` from the `foodie_level_num`. Make a scatterplot showing the two variables, and add a curve to this scatterplot that shows the predicted `rating` as a function of `foodie_level_num`. What is the test RMSE of this model?

Then, combine this model with your (optimal) model from Question 5 into a ensemble model. How does the test RMSE of the ensemble model compare to the test RMSE of each individual model?

Hint: Feel free to borrow the `RegressionEnsembler` code that I wrote. However, it will not work out of the box because the two models you are trying to combine use different variables as input. So if you use my `RegressionEnsembler`, you will have to adapt it to make it work for this problem.

```
In [21]: features = ["review.user.foodie_level_num"]

         X_train = all_reviews_df[features]
         y_train = all_reviews_df["review.rating"]

         scaler = StandardScaler()
         scaler.fit(X_train)
         X_train_sc = scaler.transform(X_train)


         model3 = KNeighborsRegressor(n_neighbors=80)
         model3.fit(X_train_sc, y_train)

         pipeline = Pipeline([
             ("scaler", scaler),
             ("model", model3)
         ])


         score = cross_val_score(pipeline, X_train, y_train,
                                 cv=8, scoring="neg_mean_squared_error").mean()

         rmse = np.sqrt(np.mean(-score))
         rmse


Out[21]: 1.4096946569859183

In [22]: def foodie_error(X_new):
             return pd.Series(model3.predict(X_new))

In [23]: X_new = pd.DataFrame()
         X_new["Foodie Rating"] = np.arange(0, 14, 1)
         X_new

         y_new_pred = foodie_error(X_new)

In [24]: all_reviews_df["review.rating"] =
             all_reviews_df["review.rating"].astype(float)

         all_reviews_df.plot.scatter("review.user.foodie_level_num",
                                     "review.rating", color="black", alpha=.2)
         y_new_pred.index = X_new
         y_new_pred.plot.line(color="red")

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd85e0905c0>
```



In [25]:
```
from sklearn.base import BaseEstimator, RegressorMixin
from sklearn.utils.validation import check_X_y, check_array, check_is_fitted
from sklearn.linear_model import LinearRegression

class RegressionEnsembler(BaseEstimator, RegressorMixin):

    def __init__(self, estimators, learn_weights=True):
        self.estimators = estimators
        self.learn_weights = learn_weights

    def fit(self, X, y):

        X, y = check_X_y(X, y)

        self.X_ = X
        self.y_ = y

        for estimator in self.estimators:
            estimator.fit(X, y)

        if self.learn_weights:
            predictions = []
            for estimator in self.estimators:
                predictions.append(estimator.predict(X))
```

```
                Y_ = np.column_stack(predictions)

                self.ensembler = LinearRegression(fit_intercept=False)
                self.ensembler.fit(Y_, y)

            return self

        def predict(self, X):
            check_is_fitted(self, ['X_', 'y_'])

            X = check_array(X)

            predictions = []
            for estimator in self.estimators:
                predictions.append(estimator.predict(X))
            Y_ = np.column_stack(predictions)

            if self.learn_weights:
                return self.ensembler.predict(Y_)
            else:
                return Y_.mean(axis=1)
```

```
In [26]: ensemble_model = RegressionEnsembler([model1, model3])

         ensemble_model.fit(X_train, y_train)
         ensemble_model.predict(X_train);
```

```
In [27]: -cross_val_score(RegressionEnsembler([model1, model2], learn_weights=False),
                          X_train, y_train,
                          cv=20, scoring="neg_mean_squared_error").mean()
```

```
Out[27]: 2.1874670566502465
```

```
In [28]: -cross_val_score(RegressionEnsembler([model1, model2], learn_weights=True),
                          X_train, y_train,
                          cv=20, scoring="neg_mean_squared_error").mean()
```

```
Out[28]: 2.1574493840185616
```

The Ensemble RMSE is actually greater than both individual models. In comparison to the 10 nearest neighbors model using the review text, the difference in RMSE's is negligible. The Ensemble RMSE is .03 points greater without learning the weights and is only .01 points better when learning the weights. However, the 80 nearest neighbors model using the foodie level number yielded an RMSE of 1.41 which is a bit less than the Ensemble model.

8 Submission Instructions

Once you are finished, follow these steps:

1. Restart the kernel and re-run this notebook from beginning to end by going to `Kernel > Restart Kernel and Run All Cells`. (If you are close to your API quota limit, do not re-run the code for Questions 1-3.)
2. If this process stops halfway through, that means there was an error. Correct the error and repeat Step 1 until the notebook runs from beginning to end.
3. Double check that there is a number next to each code cell and that these numbers are in order.

Then, submit your exam as follows:

1. Go to `File > Export Notebook As > PDF`.
2. Double check that the entire notebook, from beginning to end, is in this PDF file. (If the notebook is cut off, try first exporting the notebook to HTML and printing to PDF.)
3. Upload the PDF to PolyLearn.