MCIS6273 Data Mining (Prof. Maull) / Fall 2017 / HW2

This assignment is worth up to 20 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
20	Wednesday, Oct 18 @ Midnight	*up to* 16 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by Univerisity or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- CLEAN AND PREPARE: Work with Pandas and Scikit to preprocess and reshape a real world dataset
- SUMMARIZE AND DESCRIBE: Work with Pandas and Scikit to describe and summarize the statistical features of a real world dataset
- EXPLORE AND VISUALIZE: Work with Scikit to do exploratory data analysis on a real world dataset looking into correlation relationships and clustering

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, clone the course repository and modify the `hw2.ipynb` file in the `homework/hw2` directory. If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using Github and cloning repositories.

Turn in a copy of a `.ipynb` file, a PDF or Word Document to Blackboard with the answers to the questions labeled with the § sign.

ASSIGNMENT TASKS

(15%) CLEAN AND PREPARE: Work with Pandas and Scikit to preprocess and reshape a real world dataset

We will continue to practice preprocessing data, except this time with a much larger dataset than last time – get the coolers and heat sinks in your machines prepared. Real world data usually requires **sanitizing** of some sort – indeed it is very rare to be given a dataset that is entirely pristine unless the preprocessing was already done. *Dirty data*, so to speak, is still an important activity for data scientists, data engineerings and machine learning engineers alike. The dataset for this task is a Kaggle dataset that provides real estate transactions in New York City from 2016 and 2017 – these are *actual transactions* and this is a remarkably interesting dataset. As before, we will need to cleanse the data and prepare it so that it is in a condition that can be used by the Sklearn estimators (we'll be doing some clustering in the last part of this assignment). You are free to download the dataset (account required) or obtain it from the course Github repo. You are also

encouraged to see what others are doing with this dataset and if you learn something useful ... perhaps it will be valuable in completing your homework! NOTE: before you begin part 2 (**summarize and describe**), make sure all numeric data is forced to numeric (use `pd.to_numeric()` and set `errors='coerce'`). There are 5 tasks that will be required to do the rest of the homework (parts 2 and 3).

§ Load the dataset into a DataFrame and make sure all columns are numeric data, except for those indicated. You will need to use the `Pandas.to_numeric()` on everything **except** `NEIGHBORHOOD`, `BUILDING CLASS CATEGORY`, `TAX CLASS AT PRESENT`, `TAX CLASS AT TIME OF SALE`, `BUILDING CLASS AT TIME OF SALE` and `SALE DATE`.

§ Drop data columns as indicated:

- Drop all data values for which `SALE PRICE` is less than 1000 (e.g. `SALE PRICE>=1000`).
- Drop the entire `EASE-MENT` column.
- Drop the `ADDRESS` column. NOTE: You will need to make note of the address / index if you decide to do the bonus part (so read it before you drop, if you'd like to do the bonus).

§ Perform the following adjustments:

- Impute all `YEAR BUILT` data using the `sklearn.preprocessing.Imputer`. You can just use the median.
- also drop any properties with a `YEAR BUILT < 1600`
- Use the function

```
def convert_timedate_series(series):
    data = pd.DataFrame({ \
        'SALE MONTH': pd.to_datetime(series).dt.month, \
        'SALE DAY': pd.to_datetime(series).dt.day, \
        'SALE YEAR': pd.to_datetime(series).dt.year })

    return \
        data.columns, data
```

  to convert and split `SALE DATE` to its constituent parts. This will ease the binarization step below. You will need to append these new columns to your clean (and working) DataFrame. Try:

```
columns, data = convert_timedate_series(df_clean['SALE DATE'])
df_clean[columns] = data
```

  – and then DROP the `SALE DATE` column when this is done.

§ Final clean and save the data as indicated:

- convert all `NaN` data to `0`.
- strip and remove whitespace from any remaining columns that have text. You can use any method you want, but the `Series.str.strip()` method might be of help.
- store the un-normalized, un-binarized file as `nyc_data_clean_unscaled.csv`

§ Binarize, scale and store the scaled DataFrame:

- binarize the data – you are free to use the very convenient `Pandas.get_dummies()` or reuse any code you produced from the last exercise.

- scale the data using minmax scaling, so all data values are in the entire dataset are between 0 and 1.
- store the scaled dataset back as a CSV object (see HW0) and make sure this is turned in with your homework submission! You can name the file `nyc_data_clean_scaled.csv`.

§ **OPTIONAL BONUS WORTH UP TO 10 EXTRA CREDIT POINTS:** Get a free API key and use the GeoNames API to improve the data set, by adding a `lat` and `lon` coordinates to every property (using the address lookup). NOTE: the API limit is 20K calls for the day, so you will need to run this over several 24 hour periods between API limits (or use multiple accounts, or whatever). For the **first** student who *creates this dataset and makes it public on their Github account* (and shares it with me or your classmates), I will offer you a **total of 10 extra credit points** that will be applied to your homework total for the course. Varying levels of completeness toward that goal will earn some points less than 10.

(25%) SUMMARIZE AND DESCRIBE: Work with Pandas and Scikit to describe and summarize the statistical features of a real world dataset

Now that we have a clean dataset, we will continue to practice describing it. Here we will look at the **unscaled, unbinarized, cleaned** version from the **CLEAN AND PREPARE** step above. Answer the following questions based on that clean dataset.

§ Group the properties **by zipcode** in the clean dataset.

- What are the **top 5 zipcodes with the largest total number of properties for sale** (ignore property type for now).
- What boroughs are these zip codes in? You can inspect the dataset directly (go back to the Kaggle page for this dataset to see which code maps to the borough), use your favorite search engine to look up the zipcodes, or even use Geonames postal code lookup.

§ Using the data from above, grouped by zipcode:

- Which borough has the highest **mean** sale price of *commercial* office buildings (HINT: `'BUILDING CLASS CATEGORY'=='21 OFFICE BUILDINGS'`)?
- What is the **median sale price** of that same borough?

§ We talked about correlation in the lectures and now we're going to explore that more. Correlation can be accessed via a number of mechanisms, the primary of which is using Pandas `DataFrame.corr()` which produces a matrix representation of the correlations *among numeric variables* in your data. You can also use the `numpy.corrcoef()` which also allows you to compare single *numeric* features against one another. Both use the Pearson's correlation as we talked about in class. **Compute the correlation matrix** using either method above.

- Which features are **mostly highly correlated**?
- Do the highly correlated **features make sense** to you? Please provide a sentence of two of your reasons why or why not.

§ Filter the data to just those residential properties sold since 2005 (filter such that `RESIDENTIAL UNITS > 0 & COMMERCIAL UNITS <= 0`):

- What borough has the **largest number of new(er) properties** sold?
- What was the **mean and median price for each borough**?

§ Group properties by month and year.

- Create a line **plot of sales volume by month** (HINT: aggregate by month) for 2016 only. You may ignore 2017 properties for now. The x axis will be the month and y axis the volume (count).

(60%) EXPLORE AND VISUALIZE: Work with Scikit to do exploratory data analysis on a real world dataset looking into correlation relationships and clustering

We will finish this exercise by exploring relationships, visualizing correlations and looking at using unsupervised clustering algorithms available in Scikit.

§ In part 2 you created a correlation matrix with numbers, but this matrix can be visualized using a *heat map* visualization. You may already be familiar with heat maps (or at least seen them and not known that was what they were called), but with a correlation matrix, we can easily visualize the heatmap by converting each matrix cell (which contains the correlation between each pair of features in the data) to a color corresponding the the relative strength of the correlation. We will use open Seaborn's heatmap to do this.

- Use this boilerplate code **and turn in the heatmap for all numeric variables in your dataset**

```
%matplotlib inline
import numpy as npa
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="white")

# Compute the correlation matrix before
###
     YOUR ONE LINE OF CODE
     TO GET THE CORRELATION MATRIX GOES HERE
###

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(YOUR_CORRELATION_MATRIX_VARIABLE_HERE,
            mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, annot=True,
            linewidths=.5, cbar_kws={"shrink": .5})
```

§ Cluster the **scaled, binarized, clean** data using the Expectation Maximization algorithm via `sklearn.mixture.GaussianMixture`. You can use the defaults except change the `n_inits` parameter to 10 and set the number of clusters to 5. We will explore the clusters in your results – please provide *evidence* for all of your answers in the form of the actual data from your clusters.

To retrieve the original values of the data, you have two choices. You can simply unscale the data (use minmax in *reverse*) or you can keep track of the index and reference the data from the unscaled clean dataset (both datasets will be the same in terms of their indices). You will need to perform this trick on *both* clustering tasks.

- What are the **sizes of each of the 5 clusters** (in total count per cluster)?

- What is the **most frequent property type** (e.g. `BUILDING CLASS CATEGORY`) in your largest cluster?
- What about the **average prices in each cluster**?
- What is the **ratio of `SALE PRICE` to `GROSS SQUARE FOOTAGE`** (use the unscaled data for the ratio)?

§ Cluster the **scaled, binarized, clean** data using Agglomerative clustering with `sklearn.cluster.AgglomerativeClusteri`
Set the `n_clusters=5`, default linkage to `linkage='complete'` and the `affinity='cosine'`.

- What is the **median gross square footage** in each cluster?
- Compare the **most frequent property types** with those found with the EM algorithm. What are the differences?
- What is the **ratio of `RESIDENTIAL UNITS` to `COMMERCIAL UNITS`** per cluster?

§ Please turn in all your code and notebooks with your exploration and output.