Introduction

Welcome to **CS148 - Data Science Fundamentals!** As we're planning to move through topics aggressively in this course, to start out, we'll look to do an end-to-end walkthrough of a datascience project, and then ask you to replicate the code yourself for a new dataset.

Please note: We don't expect you to fully grasp everything happening here in either code or theory. This content will be reviewed throughout the quarter. Rather we hope that by giving you the full perspective on a data science project it will better help to contextualize the pieces as they're covered in class

In that spirit, we will first work through an example project from end to end to give you a feel for the steps involved.

Here are the main steps:

1. Get the data
2. Visualize the data for insights
3. Preprocess the data for your machine learning algorithm
4. Select a machine learning model and train it
5. Evaluate its performance

Working with Real Data

It is best to experiment with real-data as opposed to aritifical datasets.

There are many different open datasets depending on the type of problems you might be interested in!

Here are a few data repositories you could check out:

- UCI Datasets
- Kaggle Datasets
- AWS Datasets

Below we will run through an California Housing example collected from the 1990's.

Setup

[ ] ↳ *已隐藏 2 个单元格*

Intro to Data Exploration Using Pandas

[ ] ↳ *已隐藏 13 个单元格*

Let's start visualizing the dataset

[ ] ↳ *已隐藏 17 个单元格*

Preparing Dastaset for ML

[ ] ↳ *已隐藏 22 个单元格*

TODO: Applying the end-end ML steps to a different dataset.

Ok now it's time to get to work! We will apply what we've learnt to another dataset (airbnb dataset). For this project we will attempt to **predict the airbnb rental price based on other features in our given dataset.**

Visualizing Data

Load the data + statistics

Let's do the following set of tasks to get us warmed up:

- load the dataset

- display the first few rows of the data
- drop the following columns: name, host_id, host_name, last_review, neighbourhood
- display a summary of the statistics of the loaded data

```
import pandas as pd
# load the dataset
airbnb = pd.read_csv('AB_NYC_2019.csv')
# display the first few rows of the data
airbnb.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mi
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	

```
# drop the following columns: name, host_id, host_name, last_review, neighbourhood
airbnb_data = airbnb.drop(["name","host_id","host_name","last_review","neighbourhood"],axis=1)
# display a summary of the statistics of the loaded data
airbnb_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   neighbourhood_group             48895 non-null  object
 2   latitude                        48895 non-null  float64
 3   longitude                       48895 non-null  float64
 4   room_type                       48895 non-null  object
 5   price                           48895 non-null  int64
 6   minimum_nights                  48895 non-null  int64
 7   number_of_reviews               48895 non-null  int64
 8   reviews_per_month               38843 non-null  float64
 9   calculated_host_listings_count  48895 non-null  int64
 10  availability_365                48895 non-null  int64
dtypes: float64(3), int64(6), object(2)
memory usage: 4.1+ MB
```

Some Basic Visualizations

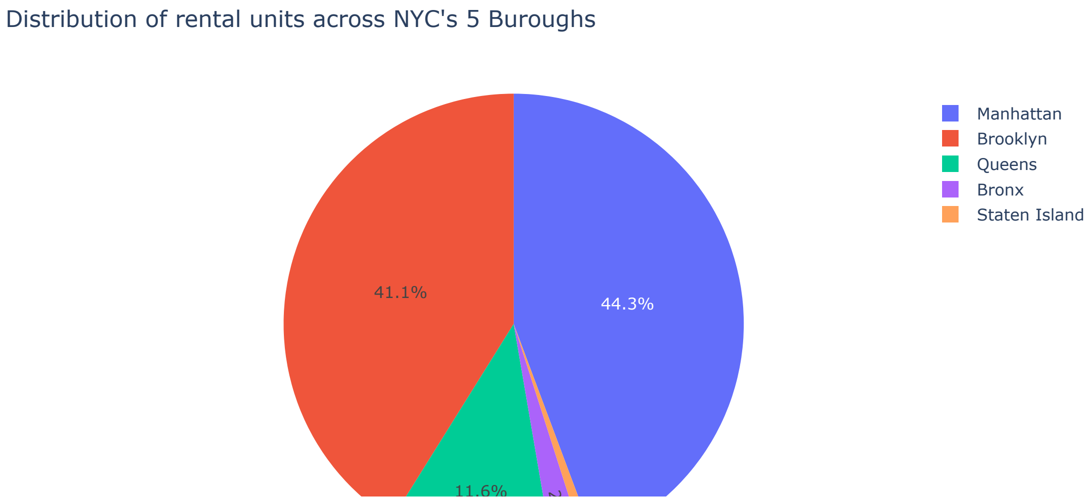
Let's try another popular python graphics library: Plotly.

You can find documentation and all the examples you'll need here: [Plotly Documentation]

Let's start out by getting a better feel for the distribution of rentals in the market.

Generate a pie chart showing the distribution of rental units across NYC's 5 Buroughs (`neighbourhood_groups` in the dataset)

```
import plotly.express as px
# set pie chart
fig = px.pie(airbnb_data, names='neighbourhood_group', title='Distribution of rental units across NYC\'s 5 Buroughs')
fig.show()
```



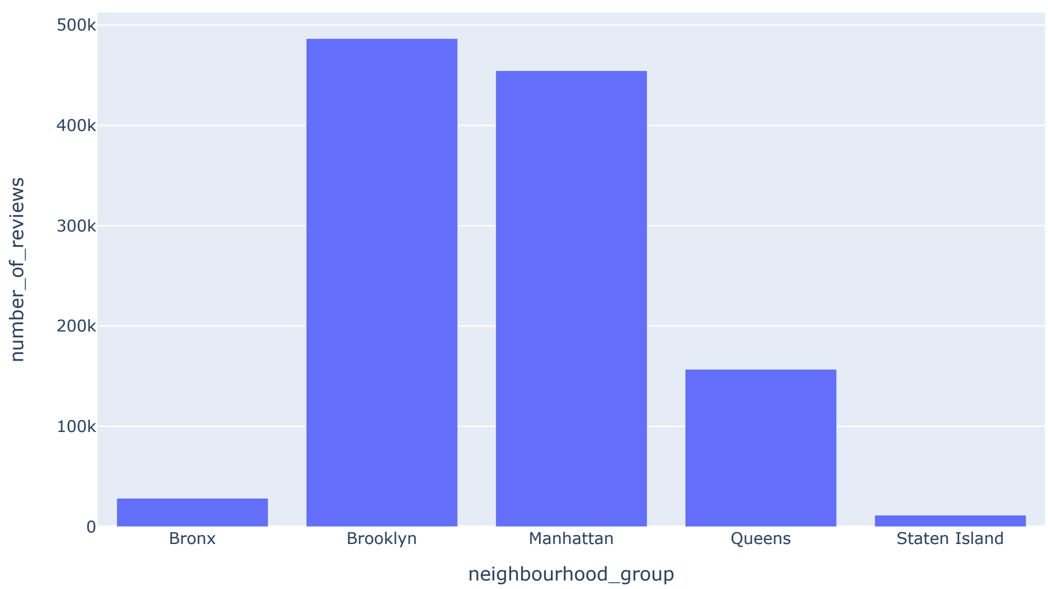
Plot the total number_of_reviews per neighbourhood_group

We now want to see the total number of reviews left for each neighborhood group in the form of a Bar Chart (where the X-axis is the neighbourhood group and the Y-axis is a count of review.

This is a two step process:

1. You'll have to sum up the reviews per neighbourhood group **(hint! try using the groupby function)**
2. Then use Plotly to generate the graph

```
# count total reviews in groups
total_reviews = airbnb_data.groupby("neighbourhood_group").agg("sum")
# set bar chart
fig = px.bar(total_reviews, x=total_reviews.index, y='number_of_reviews')
fig.show()
```



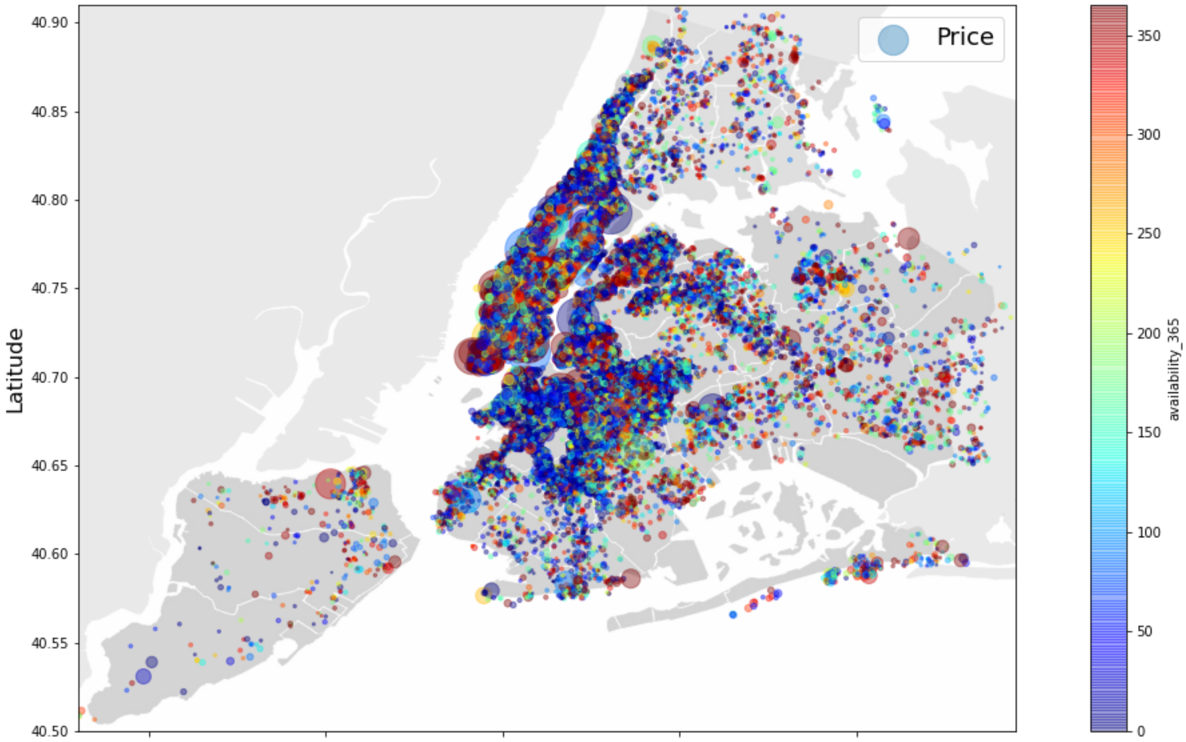
Plot a map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

For reference you can use the Matplotlib code above to replicate this graph here.

```
import matplotlib.image as mpimg

# load image
nyc_img=mpimg.imread('nyc.png')

ax = airbnb_data.plot(kind="scatter", x="longitude", y="latitude", figsize=(20,10),
                      s = airbnb_data["price"]/10, label="Price",
                      c = "availability_365", cmap=plt.get_cmap("jet"), colorbar=True, alpha=0.4)
plt.imshow(nyc_img, extent=[-74.24, -73.71, 40.50, 40.91],alpha=0.3, cmap=plt.get_cmap("jet"))
plt.ylabel("Latitude", fontsize=16)
plt.xlabel("Longitude", fontsize=16)
plt.legend(fontsize=18)
plt.show()
```



Now try to recreate this plot using Plotly's Scatterplot functionality. Note that the increased interactivity of the plot allows for some very cool functionality

```
import plotly.express as px
fig = px.scatter(airbnb_data, x="longitude", y="latitude", color="availability_365",
                  size="price", hover_data=['neighbourhood_group'])
fig.show()
```

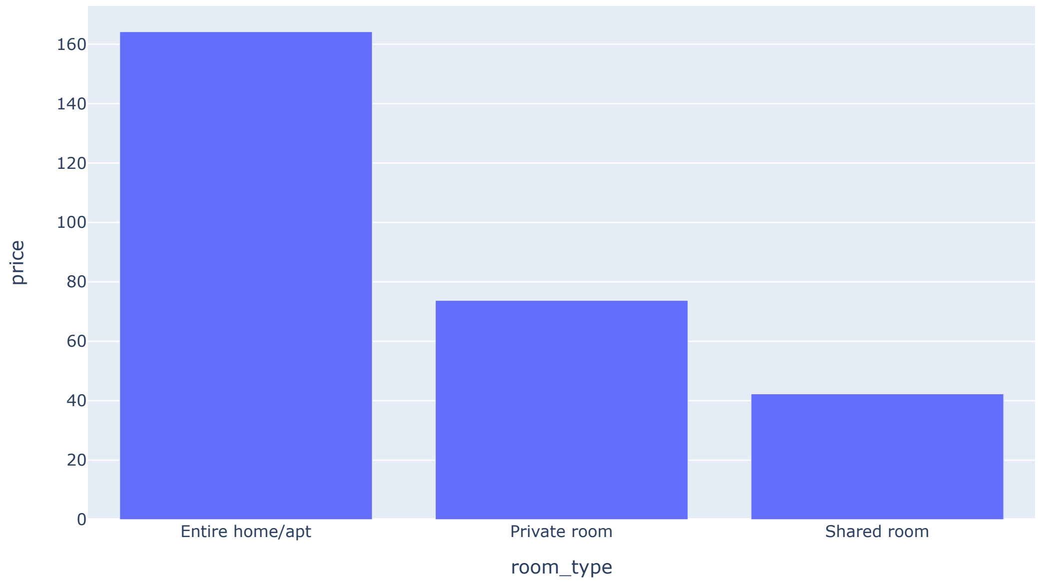
Use Plotly to plot the average price of room types in Brooklyn who have at least 10 Reviews.

Like with the previous example you'll have to do a little bit of data engineering before you actually generate the plot.

Generally I'd recommend the following series of steps:

1. Filter the data by neighborhood group and number of reviews to arrive at the subset of data relevant to this graph.
2. Groupby the room type
3. Take the mean of the price for each roomtype group
4. FINALLY (seriously!?!?) plot the result

```
# Filter the data by neighborhood group and number of reviews to arrive at the subset of data relevant t
airbnb_avg = airbnb.where((airbnb["neighbourhood_group"] == "Brooklyn") & (airbnb["number_of_reviews"] >= 10)).grou
fig = px.bar(airbnb_avg, x=airbnb_avg.index, y='price')
fig.show()
```



Prepare the Data

Feature Engineering

Let's create a new binned feature, `price_cat` that will divide our dataset into quintiles (1-5) in terms of price level (you can choose the levels to assign)

Do a value count to check the distribution of values

```
airbnb_data["income_cat"] = pd.cut(airbnb["price"], bins=[0., 50., 100., 150., 200., np.inf], labels=[1, 2, 3, 4, 5] )
airbnb_data["income_cat"].value_counts()
```

```
2    17367
3    10029
5     8384
4     6554
1     6550
Name: income_cat, dtype: int64
```

Now engineer at least one new feature.

```
# new feature
# minimum cost for each visitor
```

```
airbnb_data["minimum_cost"] = airbnb_data["minimum_nights"] * airbnb_data["price"]
airbnb_data["minimum_cost"].head()
```

```
0    149
1    225
2    450
3     89
4    800
Name: minimum_cost, dtype: int64
```

Data Imputation

Determine if there are any null-values and if there are impute them.

```
# check which values is null
airbnb_data[airbnb_data.isnull().any(axis=1)].head()
# fill 0 for reviews per month
airbnb_data["reviews_per_month"].fillna(0, inplace=True)
```

Numeric Conversions

Finally, review what features in your dataset are non-numeric and convert them.

```
from sklearn.preprocessing import LabelEncoder

# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
airbnb_data['income_cat'] = labelencoder.fit_transform(airbnb_data['income_cat'])
airbnb_data['neighbourhood_group'] = labelencoder.fit_transform(airbnb_data['neighbourhood_group'])
airbnb_data['room_type'] = labelencoder.fit_transform(airbnb_data['room_type'])
airbnb_data.head()
```

	id	neighbourhood_group	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	c
0	2539	1	40.64749	-73.97237	1	149	1	9	0.21	
1	2595	2	40.75362	-73.98377	0	225	1	45	0.38	
2	3647	2	40.80902	-73.94190	1	150	3	0	0.00	
3	3831	1	40.68514	-73.95976	0	89	1	270	4.64	
4	5022	2	40.79851	-73.94399	0	80	10	9	0.10	

Prepare data for Machine Learning

Set aside 20% of the data as test test (80% train, 20% test).

Using our `StratifiedShuffleSplit` function example from above, let's split our data into a 80/20 Training/Testing split using `neighbourhood_group` to partition the dataset

```
from sklearn.model_selection import StratifiedShuffleSplit
# let's first start by creating our train and test sets
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(airbnb_data, airbnb_data["neighbourhood_group"]):
        train_set = airbnb_data.loc[train_index]
        test_set = airbnb_data.loc[test_index]
```

Finally, remove your labels `price` from your testing and training cohorts, and create separate label features.

```
# drop labels for training set features
# the input to the model should not contain the true label
airbnb_training = train_set.drop("price", axis=1)
airbnb_labels = train_set["price"].copy()
airbnb_testing = test_set.drop("price", axis=1)
airbnb_test_labels = test_set["price"].copy()
```

Fit a model of your choice

The task is to predict the price, you could refer to the housing example on how to train and evaluate your model using **MSE**. Provide both **test and train set MSE values**.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_reg = LinearRegression()
lin_reg.fit(airbnb_training, airbnb_labels)

# let's try our model on a few testing instances
data = airbnb_testing.iloc[:5]
labels = airbnb_test_labels.iloc[:5]

print("Predictions:", lin_reg.predict(data))
print("Actual labels:", list(labels))
```

```
Predictions: [164.9075733  -23.19648225 227.88475143  58.8144876  154.35592636]
Actual labels: [120, 35, 200, 75, 150]
```

```
# MSE of testing set
preds_test = lin_reg.predict(airbnb_testing)
mse_test = mean_squared_error(airbnb_test_labels, preds_test)
mse_test
```

```
38331.46265848862
```

```
# MSE of training set
preds_training = lin_reg.predict(airbnb_training)
mse_training = mean_squared_error(airbnb_labels, preds_training)
mse_training
```

```
33827.50638958112
```