January 26, 2024

C964: Computer Science Capstone

William Jackson

Task 2 parts A, B, C and D

Part A: Letter of Transmittal

A.1 Letter of Transmittal Requirements

24 January 2024

CEO Jonathan Weizman

Weizman Housing Consultancy

153 W 34th St Apt 204., New York, NY 10001

Dear Mr Weizman,

I am writing to propose the development of a program that can address the difficulties we have been having with providing timely and accurate rental predictions to our clients. As you know, we are currently not to give a single price estimate in response to client requirements regarding size, the number of bedrooms, the number of bathrooms, and other such criteria. Instead, we have returned qualified versions of "it depends". This approach has also yielded inconsistent answers to our clients depending on the assigned agent, because our data on housing listings is spread across different spreadsheets and because each agent performs its online search on housing sites to arrive at a qualified answer. The data team in particular has been overburdened as they both need to assist agents with queries, update and maintain databases and spreadsheets as listings are removed or added to the market, and manually create reports working with complex statistical tools.

To address these issues, I propose a solution incorporating machine learning to provide a single price prediction with the caveat that it can be slightly higher or lower depending on other factors (such as the inclusion of parking garage space, skyline views, etc.). Such a machine-

learning-based goal is achievable by applying an algorithm that maps price patterns to client requirements to generate a prediction based on all the data we can feed it. Agents can learn to utilize the user-friendly model to deliver answers instantly to clients, without needing to refer to the data team for aid or spend time gathering their information via external hosting sites. The estimates the company provides to clients will in this way also be consistent because each agent will be applying the same model. The envisioned solution also includes an automatic import of the latest data from the Zillow database every day, which frees the data team from needing to maintain this list while providing the company with a single, common source for all information related to the current available rental stock in the city. I should also add that once the program is developed and implemented, it will only improve over time as the model learns from its previous correct and incorrect guesses and as further filters are added (such as pet allowance or wheelchair accessibility) to refine the predictions. I believe that a perfect model to meet all our needs needs to begin with a basic version that can be expanded upon.

The proposed solution benefits the company by reducing the time needed to meet our client's needs, reducing labor costs, and generating added business. It saves us time because no time would be needed to maintain our database and spreadsheets, since currently, our data and software teams are spending valuable time manually updating these data files. Our agents will save time by spending less time awaiting predictions from the data team, and the data team will spend less time meeting these agent requests. Perhaps most importantly, this solution will generate more business for the company. The program will allow agents to provide useful, single-price predictions to client queries in a matter of seconds. This level of service will help us retain our current clientele and attract further clients as this program will give us a tactical advantage in the competitive industry we are in.

The total cost of this proposal is estimated to be $25.000. Labor costs will constitute the majority of the costs as it would require a machine learning engineer, a backend developer, and a frontend developer all devoting two whole weeks to the project. At an average hourly salary of $75, two weeks will cost $18.000 in labor costs. In addition, presuming that computers will needed, a computer with accessories costing $2000 per person will equal $6000 in personal

hardware costs. Running an app server to host the program will cost an estimated $500 monthly in electricity, hardware depreciation, and internet usage. A database server will be less taxed and is therefore estimated to cost half that of an app server at $250. Lastly, while most of the software tools used are open-source and therefore free, it is possible that some proprietary tools will be needed and $250 is allocated accordingly to software. Once the program is deployed, there will be a recurring monthly expenditure of $750 for the app and database servers.

Item	Cost
Labor costs	$18.000
Computers	$6.000
App server	$500
Database server	$250
Software	$250
Total	**$25.000**

The timeline for the project is two weeks. At the end of the first week, I expect a working version of the program to be ready for demonstration. After the second week, the program will have been refined, tested, and deployed to an app server for internal use by every employee of the company.

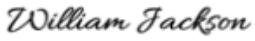
You may be aware that many machine learning projects have raised ethical concerns. I can assure you that there will be no ethical issues involved in this program as the data used will solely be public rental listings from the Zillow database.

I would like to volunteer to lead this project as the backend developer. I have several years of experience in the field, both from Weizman Housing Consultancy and from my previous employer. During my time at Weizman, I've gained a keen understanding of the needs of our clients and our agents, and I will apply these insights as we develop a user-friendly program

that addresses their requirements. I've also recently attained a degree in computer science whereby I gained skills in machine learning, frontend development, databases, and software engineering that will be useful for leading a project that encompasses these features. I have also accumulated several certifications in the field to validate my skills, including certifications in Linux and IT Service Management that provided the necessary knowledge to configure servers and lead teams.

I await your response and hope that you will regard this idea favorably.

Sincerely,



William Jackson, Sr Software Engineer

Part B: Project Proposal Plan

B.1 Project Summary

The Problem

The overall goal of the project is to provide clients with the most requested data point – estimated rental expenditure given size, number of bedrooms, and number of bathrooms. Currently, our agents are issuing qualified “it depends” estimates as answers to our clients, often with examples of listings as illustrations. This approach is both unsatisfactory and time-consuming and will be corrected by this project. T

The project will also solve the problem of scattered data sources and inconsistent service to our clients. Currently, our analysts are relying on third-party search engines, internal spreadsheets, and the ad-hoc analyses prepared by the data team. This has led to inconsistent approaches and rental expenditure predictions for our clients based on the intuition, time availability, and data source preference of the individual agent. The project will solve these issues by delivering a single database holding all information related to rental listings, and a single model that all agents can query for price predictions - ensuring consistency across agents. Relatedly, the IT department will no longer need to manually update the database with apartment listings to prevent the data from becoming stale. A script will run on the app server every night during minimal system load that pulls all current listings from the Zillow API (the largest rental listings site in the country) to populate our database that the machine learning model immediately trains on to update its predictive algorithm based on price patterns.

An added benefit of the project is that it removes the current bottleneck of awaiting responses from an overburdened data team. Agents have too often relied on deferments to the data team to retrieve answers to client queries. As this mode of communication is usually asynchronous, the result has been long response times to our clients as the chain of client -> agent -> data team -> agent -> client has at times spanned more than a week. The price predictor model

delivered by the project is meant to be used by agents directly, thus alleviating the burden on our data analysts.

Client needs in relation to the problem

Weizman Housing Consultancy has a diverse group of clients. There are city social services that must budget for housing expenditures for people within their care, companies with branches in the city that internally relocate their employees, financial advisors, and individuals moving to or within the city. However, their needs are broadly the same: they need to know what they should expect to pay for housing in the city given their requirements. They have a follow-up need of needing to know which requirements are relatively expensive or cheap to satisfy, for example, if adding an extra bedroom or bathroom to their search renders the city unaffordable for them. Another need is the ability to determine if the price of a given listing is reasonable, given its attributes, to avoid paying above-market rate prices for an apartment. The proposed program will solve all three identified needs.

Deliverables

The project will yield three deliverables by the end of the 2-week timeline. Most importantly, a working program that can predict rental prices based on size, number of bedrooms, and number of bathrooms will be made available for use by the agents working directly with clients. This program can be expanded with further independent variables in future sprints. A related deliverable is a user guide to the program with separate sections for access by office-based desktops, laptops, and mobile phones. The third deliverable is an updated database with all the rental listings in NYC pulled from the Zillow API each night. While this deliverable is utilized by the program, an additional benefit is that this database can be used for other purposes by different teams for example by the data team to prepare business analyses.

How the app benefits the client

The app will benefit the client by meeting the three needs identified in the "The Problem" section. Agents can use the program to provide a single expense estimate given the client's requirements for size, bedrooms, and bathrooms.

The ability to toggle requirements to measure their influence on the price is also achieved because a prediction will be generated within a second, which means that requirements can easily be modified to execute the model again for a new prediction with the modified input data.

Lastly, the agent can input the data from a given listing into the model to generate an expected price, which the agent can use to advise clients as to whether the price is above, at, or below the market rate. This will help our agents recommend listings that prove a good value for the price.

B.2 Data Summary

Raw data source

Zillow will be the source of the raw data once the project is implemented because this site is the source of the largest site for rental listings in the country, and they provide their API free of charge to any user that accepts certain use conditions. The data will be collected via a nightly CRON job script that makes an HTTP request to the Zillow API at the time of minimal server and database load. This data will then be handled and reformatted to populate our internal database.

Lifecycle Data Management

The data from the Zillow database will be used from the very beginning so that the design and development will be based on the same data that will be used in the production phase to ensure a smooth transition when it is implemented.

During the design phase, the available data will be evaluated with a focus on the available attributes (size, pet allowance, parking availability et.al) to confirm that the three envisioned independent variables (size, number of bedrooms, number of bathrooms) are sufficient for the initial version of the program. Other possible variables will be discussed and tested during development if they appear relevant.

During development, a prototype of the model will be trained on the data to test the model's accuracy rate. The dataset will be split into a training set and a testing set so that the model can train on the training set and be tested on the testing set. If the accuracy score is no better than random selection, then the model will need to be revamped. Additional variables identified in the design phase will be added to the model to see if that increases the predictive power of the model, in which case they can be permanently added.

During the maintenance, the data will be requested nightly from the Zillow API to repopulate the database at a time of minimal user load. This newly updated data will immediately be retrieved via a script that uses it to retrain the model so that the model's predictions are based on the latest data correlations and patterns in the housing market.

The app and database servers' caches will also accumulate data throughout the day as they will retain all queries and query responses before being cleared at the end of the day. This is done to preserve computational resources during peak load times by reusing the first response for a particular query throughout the day.

How the data meets the project needs

The Zillow database is the largest database containing rental listings in the city with approximately 10K listings at a given time. It's therefore a comprehensive database that covers a very large portion of the listings in the city, which means that the model will be trained and fitted to real-time data that can be expected to correlate closely with the complete population of current listings. Currently, only four values will be used from each listing record (size, number of bedrooms, number of bathrooms, price), but if any of these values are missing from a record, that record will be discarded from the training set.

Ethical or legal concerns

There are no ethical concerns as all data relates to publicly available rental listings. From those listings, all identifiable information or names involved (possibly an owner or a real estate agent) will be discarded as only the values related to the apartment's features will be retained and written to the database. There are likewise no legal concerns as Zillow provides its data sets free of charge to outside users via the Zillow API.

B.3 Implementation

Methodology

An Agile methodology will be applied for this project. The advantage of this approach is its flexibility in the face of changing requirements while also being a good fit for small teams. The two-week timeline to develop and implement the solution corresponds to the duration of a sprint wherein every phase, from requirements gathering to design, development, and testing will take place. The requirements gathering phase will include the end-user, and the agents, who will be invited to propose feature ideas and discuss their needs. The design and implementation phases will be handled by the 3-person development team.

Because it's an internal product, the testing phase can be lighter than usual, with a single QA tester testing the app servers' endpoints and a customer agent tasked with user acceptance testing. Once implemented, it will be further "tested" in the field by the agents, which can lead to ideas for refinements or new requirements for a coming sprint.

Once the product is deemed sufficient and complete, it will revert to the maintenance phase where any bugs will be addressed via bug fix tickets.

Implementation plan

The whole development process is projected to last two weeks or ten working days.

The first two days will be devoted to requirements gathering and design, which will entail meetings with all stakeholders, including end-user agents, the data team, and the CTO to finalize requirements.

During the following three days (days 3-5), the three team members will split and work on their assigned features. The frontend developer will develop the user interface, the backend developer will launch the app server and database, and the machine learning engineer will develop the predictive algorithm.

On day six, the features developed during the previous three days will be integrated into a functioning app.

On the seventh and eighth day, the integrated app will be refined to ensure that all parts communicate properly and send data in compatible formats.

The ninth day will be devoted to unit, integration, system, and user acceptance testing. The three team members will conduct unit tests of their own developed features, a QA tester will conduct the integration and system tests, and a customer agent will perform a user acceptance test.

On the tenth and last day, the program will be deployed into the production environment.

B.4 Timeline

Deliverable	Assignment	Duration	Start date	End date
Requirements gathering	Whole team, end users, CTO	2 days	2/5/2024	2/6/2024
User interface	Frontend developer	3 days	2/7/2024	2/9/2024
Database	Backend developer	1 day	2/7/2024	2/7/2024
App server	Backend developer	2 days	2/8/2024	2/9/2024
Machine Learning algorithm	Machine learning engineer	3 days	2/7/2024	2/9/2024
Integration into a complete app	Whole team	1 day	2/12/2024	2/12/2024
App refinement	Whole team	2 days	2/13/2024	2/14/2024
Testing	Whole team	1 day	2/15/2024	2/15/2024
Deployment	Whole team	1 day	2/16/2024	2/16/2024

B.5 Methodology

Verification method

In the planning phase, we will verify that all requirements have been gathered. A representative from the customer agent team and the CTO will sign off on the final requirements list.

In the design phase, the backend developer and machine learning engineer will critically evaluate the frontend developer's mock design before verifying it.

In the development phase, the team will verify the proper functioning of each other's features during the integration of the parts to ensure that they interoperate. Any incompatible data formats or incorrect URLs will be revealed and corrected at this stage. The machine learning engineer will continuously be testing and refining the predictive algorithm to verify and increase its accuracy

In the testing phase, unit, integration, system, and user acceptance tests will be performed to verify that each feature as well as the program as a whole is performant, correct, user-friendly, and meets all stated requirements.

During the implementation and maintenance phases, the end-users (customer agents) will provide further verification daily of its utility in meeting the needs of the company's clients in the form of accurate rental expenditure budgeting, listing price scrutiny, and adjustment of filters to stay within their price range.

Validation method

The accuracy and performance of the program must be validated before deployment. For accuracy, the algorithm must be at least three times as accurate as random selection. Since there are twelve price categories, random selection should on average yield eight percent accuracy. Therefore, the first version of the algorithm must attain an accuracy rate of at least twenty-five percent.

For performance, the QA tester will test that the algorithm generates its prediction within 1000 ms, which is deemed a reasonable goal.

Lastly, the QA tester and backend developer will validate that the database always contains the data from the previous night's import of the Zillow dataset.

B.6 Resources and Costs

Item	Cost
Labor costs (80 hours x $75 hourly x 3 members)	$18.000
Computers	$6.000
App server	$500
Database server	$250
Software	$250
Total	**$25.000**

The total cost of this proposal is estimated to be $25.000. Labor costs will constitute the majority of the costs as it would require a machine learning engineer, a backend developer, and a frontend developer all devoting two whole weeks to the project. At an average hourly salary of $75 and a forty-hour workweek for the three members of the team, two hundred and forty hours of work means $18.000 in labor costs. In addition, presuming that computers will be needed, a computer with accessories costing $2000 per person will equal $6000 in personal hardware costs. Running an app server to host the program will cost an estimated $500 monthly in electricity, hardware depreciation, and internet usage. A database server will be less taxed and is therefore estimated to cost half that of an app server at $250. Lastly, while most of the software tools used are open-source and therefore free, it is possible that some proprietary tools will be needed and $250 is allocated accordingly to software. Once the program is deployed, there will be a recurring monthly expenditure of $750 for the app and database servers. Since the program is hosted in-house, there will not be cloud expenses associated with it.

Part C: Application

The files for the program, dataset, and a link to the Google Colab-hosted app will be submitted.

Part D: Post-implementation Report

D.1 Solution Summary

The Problem

The overarching problem to be solved is the customer agent's inability to inability to generate a single pricepoint prediction for the cost of rental housing in New York City based on size, number of bathrooms, and number of bedrooms. Currently, agents research third-party sites and visit their spreadsheets to give an estimate based on human judgment rather than data science. Because the single pricepoint can not be generated, it follows that client requirements cannot be toggled to evaluate the savings or added costs related to adding or removing a bedroom for example. A third user request that cannot be met is the request for a verdict as the price of a given listing since there is no model that the listing's attributes can be plugged into. Additional problems associated with the current approach are inconsistent estimates and advice between agents, slow response times to client requests due to the time-consuming nature of the agent's methods, and time-consuming manual upkeep of databases and spreadsheets.

The Solution

The solution is to create a machine learning program that can rapidly generate a useful single-price prediction based on client requirements. A prerequisite for achieving this is having an updated dataset at hand that the model can train upon to maintain the highest possible predictive accuracy. A script will therefore run every night that pulls an updated dataset from Zillow's database via the Zillow API. Zillow was selected as the best source since Zillow hosts more rental apartment listings than any other site.

How the app provides the solution

The app will provide the solution to all the above-identified problems. Most importantly, agents will learn to utilize the user-friendly program to deliver rental price predictions instantly to

clients, not needing to conduct independent research to deliver unreliable estimates. Because agents can provide a single expense estimate given the client's requirements, they can easily toggle requirements and run the program again to measure the influence of slacking or increasing one or more requirements on the price that clients can expect as a result. When clients present listings that they are interested in, an agent can input the attributes of the listing as if it were user requirements to generate an expected price. If the expected price is lower than the listing price, the client can be warned that they will likely be paying above market rate if they were to select that listing. Conversely, if the model returns a lower expected price than the listing shows, the client can be advised that the given listing provides good value for the price. The problem with inconsistent and slow advice from agents will also be eradicated with the app because all agents will be using the same rapid-response model, and should thus arrive at identical price predictions when approached by a client.

D.2 Data Summary

Source of production and simulated data

In the development and production stages, the Zillow API would be queried nightly to maintain an updated, exhaustive database of apartment listings in New York City. Zillow provides its API free of charge to the public with 40 separate endpoints, and their table of rental listings for the city would be requested every night and used to repopulate the company's database. Once the process is completed, a follow-up script will be executed that trains the machine learning algorithm on the updated dataset so that the weights of each model parameter are adjusted according to the pricing of the current supply of available rental housing.

In this exploratory phase, the Zillow dataset was simulated by using a Kaggle dataset of CraigsList rental listings for the USA. The CSV dataset was downloaded and uploaded to Google Sheets which was easier to work with. All irrelevant columns were removed, such as pet allowance and smoking allowance. All rows for listings outside of NYC were likewise deleted. After these operations, 726 listings were left which I uploaded to a publicly available Github repository that I could then pull via an HTTP request in the Jupyter Notebook using the pandas

library to train and test the model. 75 percent of the 726 listings were used to train the model which was then tested on the remaining 25 percent of the 726 listings to arrive at an accuracy metric.

Lifecycle data management

During the requirements-gathering phase, I considered my data needs and possible data sources. I concluded that I would need a data set with at least 500 listings where information for price, size, and number of rooms was available for each listing. I explored publicly available data sets on Kaggle, a common source of publicly available datasets, and found a dataset of CraigsList (an open listings site) listings with the required available data that could simulate data retrieved from the Zillow API (the choice for production).

During the design phase, the available data was evaluated with a focus on the available attributes (size, pet allowance, parking availability et.al) to determine which independent variables appeared to be the best determinants of price and which variables the clients could be expected to list as requirements. Size, bedroom count, and bathroom count were selected during this process to be sufficient for the initial version of the program. During this phase, I also decided to round the price value to the nearest $500 to group the apartments into categories as this provided a neater overview and allowed for accuracy metrics for how well the machine learning algorithm predicted the price category of an apartment.

During the development phase, the model trained on the data to test the model's accuracy rate. I split the dataset into a training set and a testing set with 75 percent of the data set devoted to training and the remaining 25 percent devoted to testing. I explored adjustments to this distribution by assigning a greater and lower fraction to test, but I settled on the 25/75 percent distribution as the one that yielded the highest accuracy rate.

D.3 Machine Learning

Multinomial Logistic regression

The machine learning algorithm developed for the program uses multinomial logistic regression. Logistic regression is a classification algorithm that can be used to estimate probabilities of events happening based on the values for several independent variables. Often, logistic regression is applied to a category with a binary outcome, such as "yes" versus "no" or an event occurring or not occurring. However, it can also be applied in cases where there are multiple possible discrete outcomes, in which case it is called multinomial logistic regression, which is what is applied in this project. In this function, the event (dependent variable) must be discrete, while the independent variables can be any combination of discrete and continuous values. The multinomial logistic regression uses a logarithmic function to calculate the probability of each category occurring (a number between 0 and 1) and predicts the category with the highest probability as its outcome.

How the model was developed

I browsed the web for libraries that provided methods for computing multinomial logistic regression, and this search led me to the scikit-learn library that contains the LogisticRegression method that was a perfect fit for the task. I therefore imported the needed modules from the sci-kit library (the metrics were used later for validation).

```
from sklearn import linear_model, metrics, model_selection
```

The first step was to instantiate the logistic regression model. The solver argument is the algorithm the model should use for optimization. While the default is 'lbfgs', the documentation for scikit-learn recommends 'liblinear' for smaller datasets, and this choice did give better accuracy scores than the default 'liblinear'.

```
regression_model = linear_model.LogisticRegression(solver='liblinear')
```

The next step was to fit the model to the data. In the CSV dataset, the first three columns consisted of the independent variables and the fourth column contained the dependent variable, so I read those datasets and stored the columns in x and y variables using the pandas library.

```
import pandas as pd
```

```
url = 'https://raw.githubusercontent.com/williampj/general_study/master/housing_without_headers.csv'
headers = ["sqfeet", "beds", "baths", "price"]

df = pd.read_csv(url, names=headers)

x = df.values[:, 0:3]
y = df.values[:, 3]
```

Then using the model_selection module from the scikit-learn library, I split the dataset into a training set (75 percent of the rows) and a testing set (25 percent of the rows).

```
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.25, train_size=0.75)
```

At this point, I could train the regression model using the training sets extracted from the previous step.

```
regression_model.fit(x_train, y_train)
```

The model was now ready to make predictions. I created the user interface to generate the size, bedroom count, and bathroom count from user inputs to feed to the model this three-element array to generate the prediction that I then interpolated into a reader-friendly print statement. Because the prediction price unit is in thousands of dollars, I multiplied by 1000 to convert it to several dollars for reader friendliness.

Justification of model choice and development

I selected logistic regression because it was the perfect fit for solving the problem at hand, its simplicity, and performance.

For the program, I needed a method to determine the most probable among several discrete outcomes given a range of inputs, and multilinear logistic regression was created to address this very problem.

One advantage it has is its simplicity. As I showed in the previous section, in just a few lines of code, the model can be instantiated, fitted, and predicted as a category based on inputs. It's an advantage that the scikit-learn library has provided such a user-friendly interface for this method.

The other advantage is its performance as its performance scores are known to be high compared to other methods. I tried another method from the scikit library, svm, and its accuracy scores were consistently 0.1 lower than for logistic regression, so I retained the initial method.

D.4 Validation

Validation method

It's good practice to test logistic regression methods to evaluate their performance in the form of an accuracy metric between 0 (for no accuracy at all) and 1 (for complete accuracy).

```
def test_accuracy():
    x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.25, train_size=0.75)
    regression_model.fit(x_train, y_train)
    y_predict_regression = regression_model.predict(x_test)
    print(f'\n{metrics.accuracy_score(y_test, y_predict_regression)} was the accuracy score for this test')
```

I wrote a test_accuracy method that splits the dataset into a training set and a testing set using the regression_model module from the scikit-learn library. Then I fit the regression model to the training set and fed it to the testing set to generate a list of predicted y-values (dependent values – rental prices) for each row of independent variables. Using the "accuracy_score" method from the "metrics" module of the scikit-learn library, I could compare the predicted y-values to the actual values (rental prices) for each row by printing the returned accuracy score. The minimum performance goal of the model was that it achieved an accuracy of at least four times greater than random selection. With twelve categories, random selection would have

yielded an average accuracy of eight percent, so thirty-two percent was the minimum accuracy target for the model.

Validation results

In the user interface, one of the options (option 3) allows the user to execute the accuracy. The resulting accuracy I achieved was 0.36. While a greater number would have been desirable, it was nonetheless 4 ½ times more accurate than random selection.

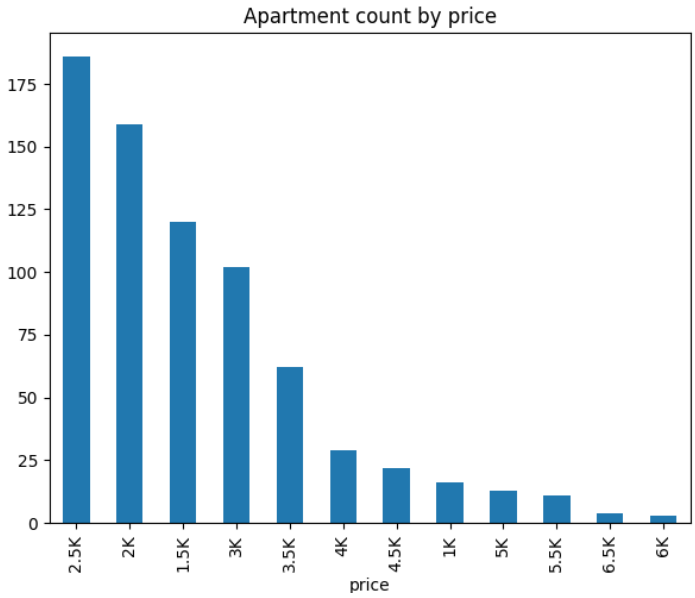
```
Please select an option between 1-3:

1: Predict rental price
2: Generate descriptive visuals of the NYC rental market
3: Test the accuracy of the ML model

3

0.3626373626373626 was the accuracy score for this test
```

Upon inspection of the dataset, I believe that splitting the dataset into boroughs would yield a higher accuracy as there appears to be significant price discrepancies between Manhattan in the one extreme and Staten Island in the other extreme, with Brooklyn and Queens generally falling in the middle. In a future iteration of the model, given a larger dataset and precise location data for each listing, it would be interesting to measure the model accuracy separately for each borough's listings.

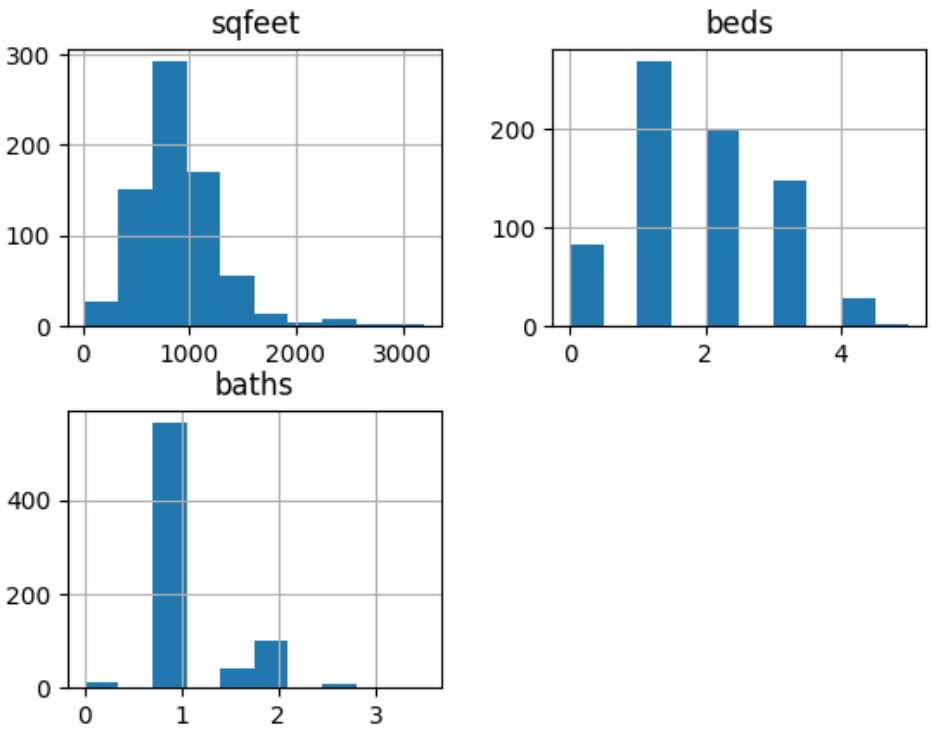
D.5 Visualizations

I provided three useful descriptive methods for the project. These can be accessed in the user interface where users have the option of requesting a price prediction (first option), requesting visuals (second option), or testing the accuracy of the model (third option). The first option was intended for the agents providing answers to client requests, while the second and third options are relevant to evaluate the data or the model. Selecting the second option will generate the following four visuals.

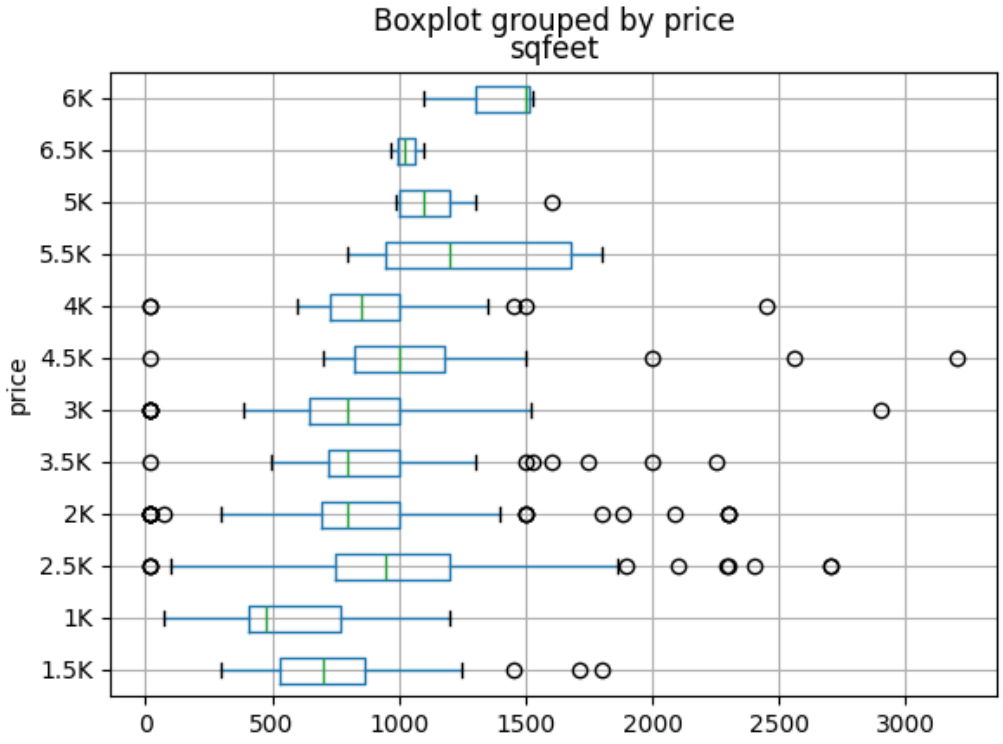
The first is a bar plot on the dependent variable (rental price) to show the distribution of apartments on the market based on price. This plot is therefore useful to gather apartment price ranges that are plentiful and which are scarce. Apartment listings priced at $2.500 followed by apartments listed at $2.000 monthly are the most numerous, while high-end apartments at $6000 or higher are the least frequent.



The second descriptive method is a histogram collection that shows the distribution among the independent variables, in other words, it illustrates which square foot sizes, bedroom sizes, and bathroom sizes are common or rare in the city. The first histogram shows that apartment sizes tend to cluster around 1000 square feet, with the largest category being apartment sizes slightly less than 1000 square feet. The second histogram shows that 1-bedroom apartments are the most frequently listed, followed by 2- and 3-bedroom apartments. The final histogram reveals that having 1 bathroom is the overwhelming standard among available listings.



The final descriptive method is a box plot between apartment size and price. This method shows the distribution, variance, and outliers of apartment sizes for each price range. Interestingly, the median size for apartments ranging between $2.500 and $4.000 are not significantly different, which indicates that there is another strong variable affecting price apart from apartment size. Upon scrutinizing the original raw data, I speculate that location is perhaps an equally strong determinant for pricing as Manhattan locations are well-known to command a significant premium to other boroughs, especially Staten Island or the Bronx. With a larger dataset controlled for borough location, I believe that the correlation between size and price would appear much stronger than is currently the case.



D. 6 User Guide

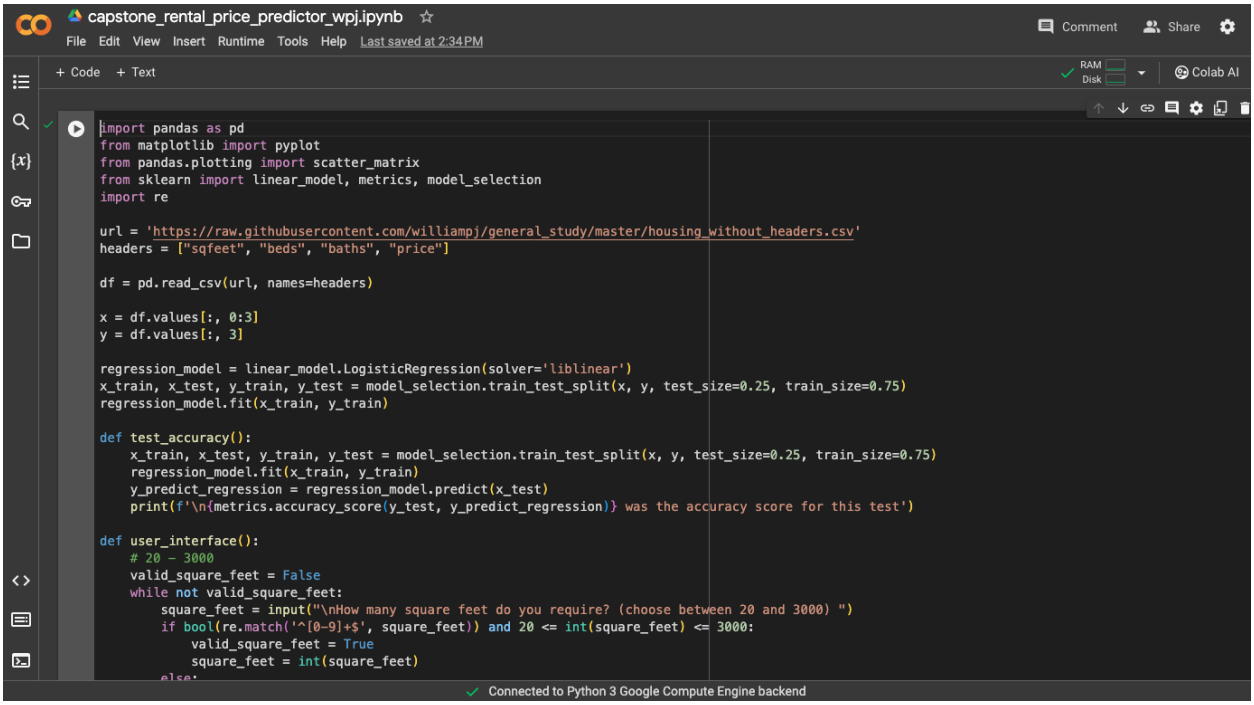
The app is hosted and accessed on Google Colab. Therefore, the only program requirement is a functioning browser (Firefox, Chrome, Safari, Opera, Edge, etc.). Because it is hosted in Google Colab, a Google account will also be needed to run it.

1. Access URL

The first step is to access the URL of the app by clicking on the following link:

https://colab.research.google.com/drive/1qTyN7k25DL23AIoDT12_vdfHevEgvmWi?authuser=0#scrollTo=sB8hgbiKcHsr

which will show the following screen:



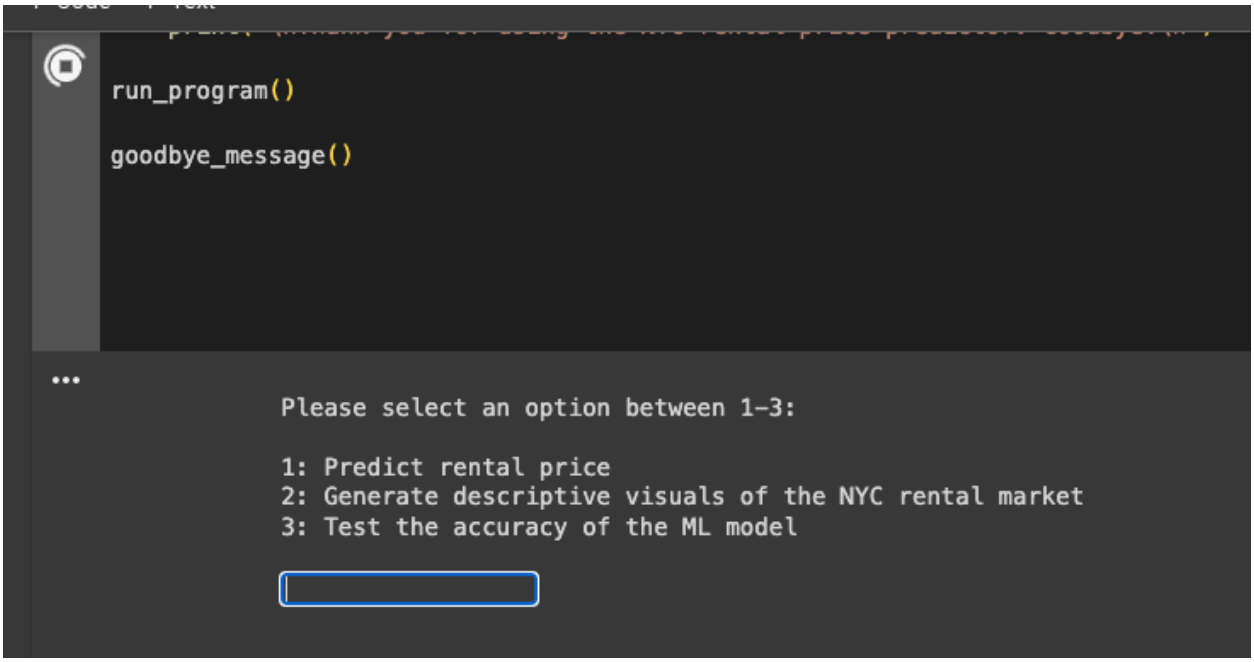
2. Run the app

The second step is to run the app, by clicking on the white-circled play button to the left of the first line in the app (*import pandas as pd*).

A Google account is required to run the app. If the user does not have one, a temporary one can be created in a couple of minutes by clicking on the blue "Sign In" button in the top right corner of the screen and following the directions.

3. Main menu

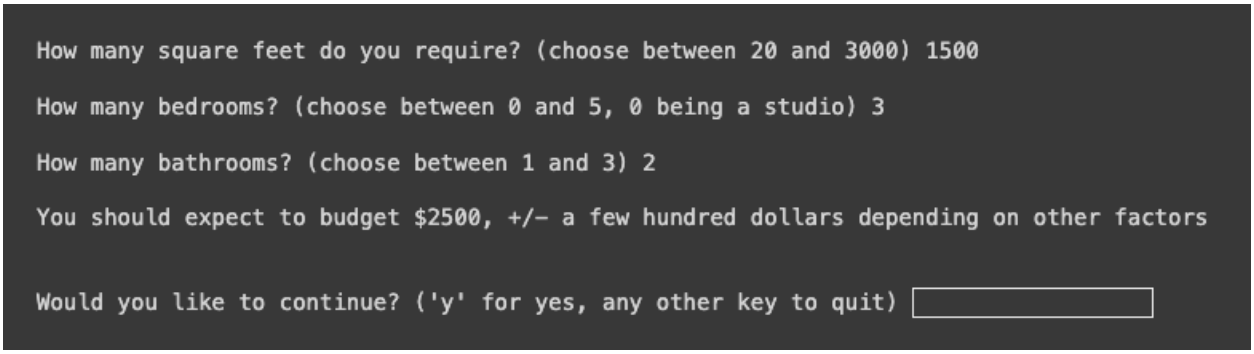
After the run button has been clicked, the user will be given a choice of three options: to predict a price, generate visuals, or test the accuracy of the model.



4.a. Run the price prediction algorithm

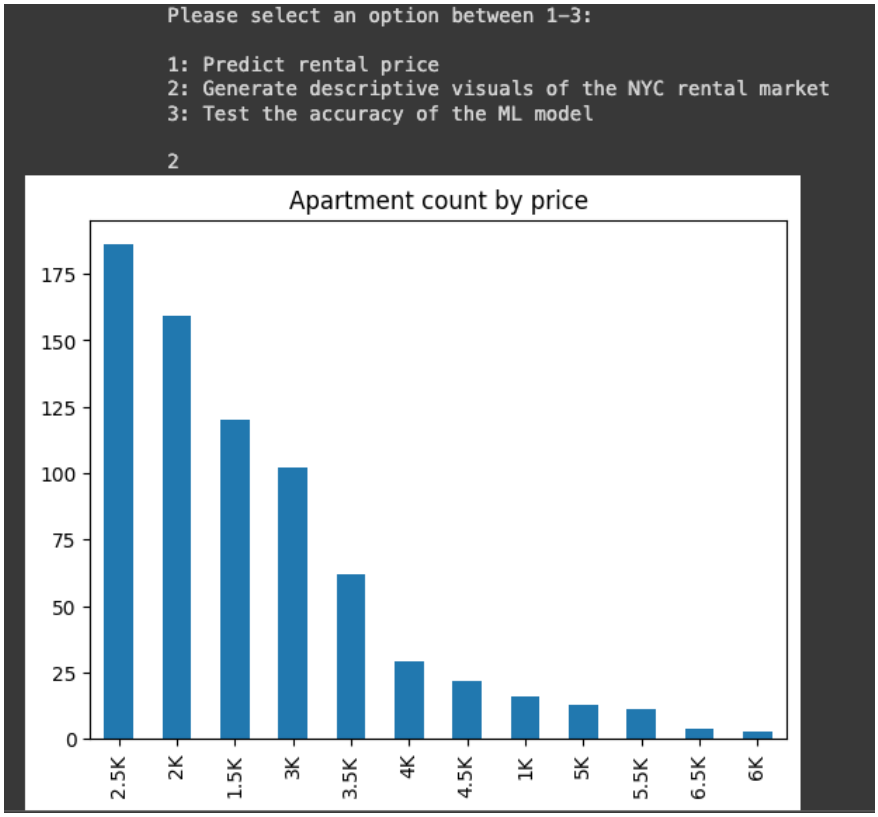
By writing '1' and pressing enter, the price predictor algorithm will be run that requires user input for size, number of bedrooms, and number of bathrooms.

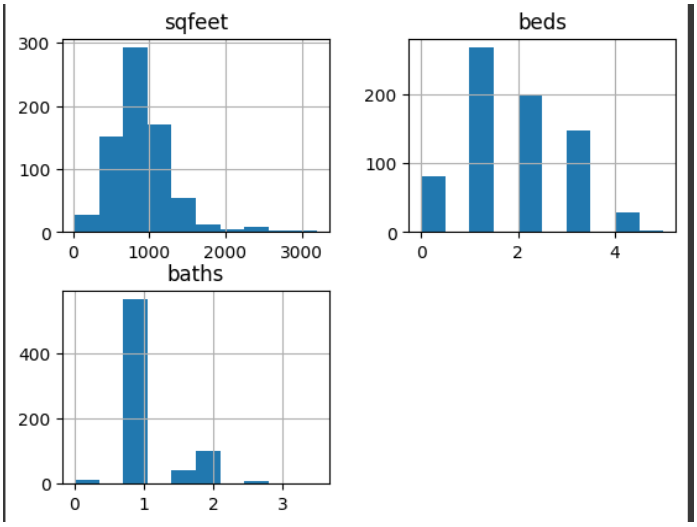
Once the three inputs have been entered within the available ranges, an expected price for a rental listing will be printed. By entering 'y' afterward, the user returns to the main menu.

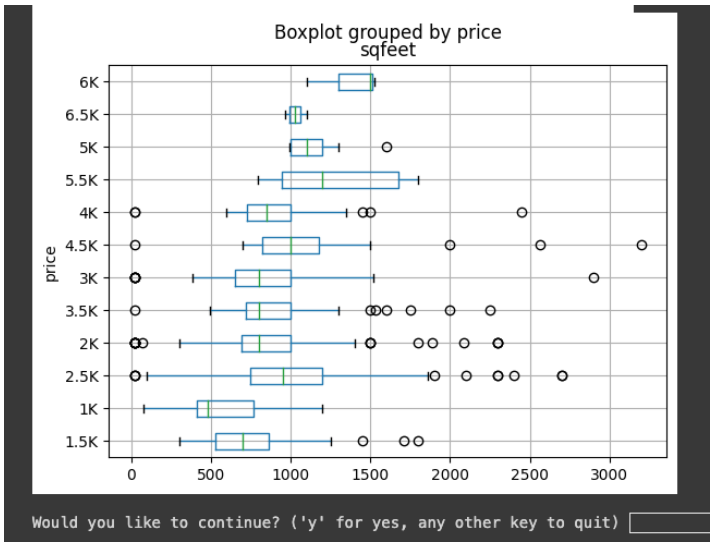


4.b. Generate visuals

If the user is interested in the graphs created for the data, the option of '2' can be selected in the main menu.

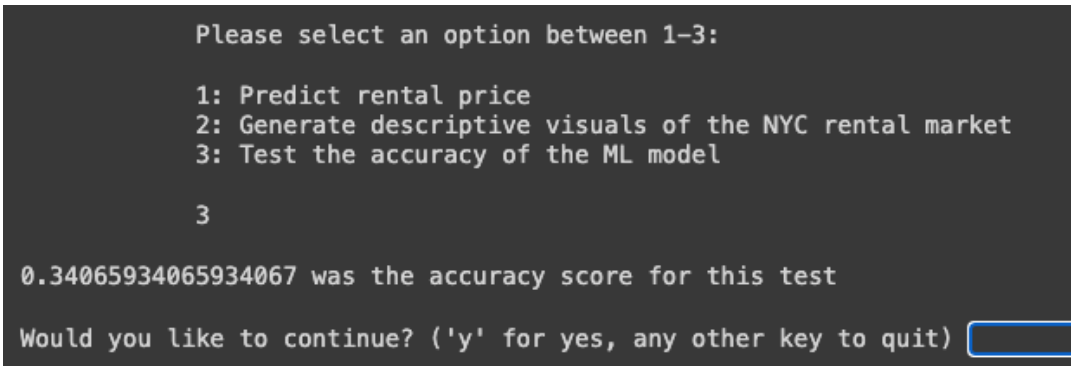



sqfeet
beds
baths


Boxplot grouped by price
sqfeet
price
Would you like to continue? ('y' for yes, any other key to quit)

4.C. Test model accuracy

If the user selects '3' in the main menu, the model will split the dataset into a training and testing sat, train on the training set, and compare predictions on the testing set to the actual values of the training set to generate an accuracy metric.



Reference Page

The only external source was the Kaggle dataset to simulate the Zillow dataset.

Bryant, Michael (2023) "Craigslist House and Apartment Rentals Dataset". Retrieved 22 January 2024 from https://www.kaggle.com/datasets/michaelbryantds/bay-area-craigslist-rentals