Realtor Insights' Assessor

Executive Summary

This feasibility report provides a detailed look into the Assessor feasibility given the course timeframe. The Assessor tool consists of four different models which attempt to aid the purchase decision of homebuyers, namely Personal Fit, Financial Opportunity, Public Perception, General Livability. The system architecture for Assessor consists of a publicly accessible web server hosted on AWS. The background and implementation of the four models are explained in the following sections. The task delegation and project management is described thereafter. Based on analyses, we concluded that most milestones are in scope, except for the General Livability model, alerting services, and the model visualization (including heatmaps).

1 Background

1.1 Motivation

Finding the right home can be a difficult task for individuals lacking experience in the real-estate market. Often, these individuals seek consultation from Realtors, who may not have the client's best interests in mind. There is a clear opportunity to empower these individuals with a real-estate assessment tool that helps them understand the neighbourhoods which best fit their needs and desires when searching for a new home.

Conducting research about neighbourhoods typically resolves to either accepting a Realtor Consultant's recommendations, surveying the area first-hand and taking the opinions of residences, or conducting research via the internet. There are some interesting tools which can help an individual conduct their own research, such as Topia's Teleport Cities [1] which gives an overview of any city across the globe to enable an individual to determine the best place to live. It provides information ranging from 'Life Quality Score' to 'Local Reviews'. It doesn't, however, provide information down to the neighbourhood granularity.

Another interesting online tool provider is Mashvisor. They offer the ‘Property Finder’ tool that helps individuals find properties based on their search criteria and investments goals [2]. In particular, it claims to leverage machine learning and artificial intelligence to find the right properties. They also offer a heat map tool that helps find the best properties from an investment perspective based on listing price, cash on return, rental income, and airbnb occupancy rate [3]. While the solutions offered by Mavishor are promising, they fail when it comes to finding the perfect home; they are more for those interested in rental properties.

1.2 Finding the Right Home

Of the numerous qualitative criteria for finding the right home, the Assessor tool addresses three criteria individually, with the remaining criteria addressed collectively.

The first criterion is the community demographics [4]. This criterion asks the question: in which neighbourhood does similar individuals reside? People with shared backgrounds may look for common characteristics in communities [4]. The Personal Fit model matches the user’s demographic information to similar residents in different neighbourhoods.

The second criterion is the local opinion [5]. This criterion asks the question: what does the general public think about the neighbourhood? Word-of-mouth information provides insight into people’s impressions of the community [5]. The Public Perception model analyzes positive and negative opinions of different neighbourhoods.

The third criterion is the value of housing [6]. This criterion asks the question: are home investments in the neighbourhood favorable? The real estate valuation in the past five to ten years may be indicative of potential trends [6]. The Financial Opportunity model forecasts future valuation based on historic data of different neighbourhoods.

The remaining criteria are chosen from Mercer’s quality of life evaluation factors [7]. Mercer analyzes living conditions according to 39 factors grouped into 10 categories [7]. These criteria ask the question: how is the quality of life in the neighbourhood? The General Livability model incorporates all of these criteria to determine the living conditions of different neighbourhoods.

The qualitative criteria are quantified in the described models, and are chosen in order to leverage our engineering knowledge to perform statistical and machine learning analysis methods.

1.3 Product Description

Assessor aims to provide a research tool for individuals interested in finding a new home that considers an individual's wants and needs at a level of granularity down to the neighborhood or district level.

A user of Assessor will visit the tool webpage where they can read over terms and conditions or begin the search. When starting the search, the user will input several characteristic parameters about themselves such as age, net family income, ethnic heritage, partner status, number of children, etc.; as well as characteristics about their desired property such as investment horizon, proximity to schools, and so on (the specifics of each of these inputs is defined in section 2.1.2 through 2.1.5). These input parameters will then be categorized by model and sent to their corresponding model.

There are 4 such models: Personal Fit, which classifies the individual with respect to demographics of each district; Public Perception, which determines the general sentiment about each district based on news and social media; Financial Opportunity, which provides a forecasted valuation of the district based on historical data; and General Liveability, which classifies the district in comparison to the ideal district, which is determined by characteristics of districts in cities with particularly high ratings in livability. Each of these models is discussed in greater detail in section 2.1.1 through 2.1.5.

When each model has completed processing, results are sent back to the individual, and displayed in a heat-map. There are 5 heat-maps that the user may click through; one per model and an aggregate heat-map that weighs the results for each district from the 4 models. The user may adjust these weights in real-time; for example, they may set financial opportunity to 70% and the remaining 3 categories to 10% each.

Overall, Assessor aims to help the individual make an informed purchase decision by aiding the research of their next home's location.

2 Feasibility of Assessor

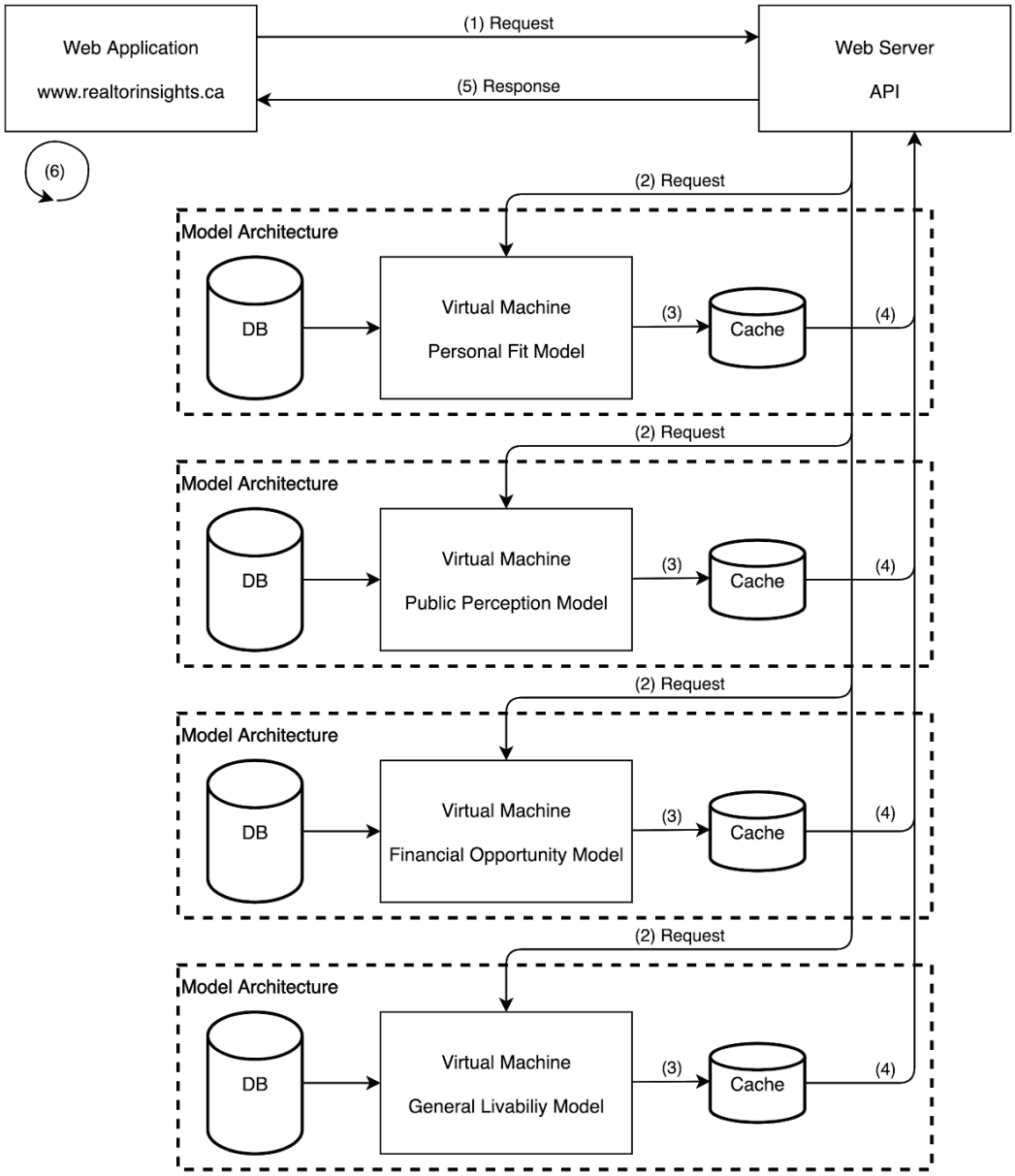
For the purposes of analyzing the feasibility of Assessor, the TELOS feasibility model is leveraged [8]. TELOS is a mnemonic for Technology, Economic, Legal, Operational, and Scheduling feasibility, and it helps determine the specific needs to implement a particular system. In Technological feasibility, an analysis is conducted to determine if the technology exists to implement the described solution; Economic feasibility determines the costs associated with the development and implementation; Legal feasibility analyses conflicts between the system and legal requirements; Operational feasibility determines whether there exists adequate support (a set of responsibles) to develop and implement the system; and Scheduling feasibility sets out the product roadmap including milestones and time-frames for each.

2.1 Technology

In the following sections the feasibility of each of the system's technical components is analyzed and specifics on implementation are determined.

2.1.1 System Architecture

Since Assessor is intended to be readily accessed by those interested in researching neighbourhoods, the application will be hosted on a publicly accessible web server. Figure 1 depicts the system architecture for Assessor. Each component is discussed in detail in the proceeding sections.



(1) Requests are sent by the web application to the server containing information about the variable parameters per model
(2) The server dispatches the requests to their respective models for processing
(3) The model pulls data from its database and returns its results to a results cache which holds the probability of fit per neighbourhood
(4) The cache then sends results back to the server via the API
(5) The server responds to each request with the results obtained in (3)
(6) The web application renders each model's results into their corresponding heat-map, and calculates an overall heat-map based on model type weighting which the user may adjust in real time

Figure 1: System Architecture Diagram for Assessor

For Assessor, the entire application will be deployed onto Amazon's AWS stack. Amazon provides a multitude of services and hosting options which makes it easy for developing complex applications. In particular, AWS provides the following benefits [9][10]:

- A pay-as-you-go fee structure which means we only pay for the web traffic to the site (S3) or computation time (EC2);
- Highly reliable with a quoted durability (availability) of 99.999999999%;
- Secure with many configuration options, like VPC which can restrict particular assets to a private network
- And automatic load balancing via Cloudfront

2.1.1.1 Web Application

The web application serves as the main point of interaction for users of Assessor. When a user visits the website URL, realtorinsights.ca, they will be greeted with a welcome page which points to several other pages hosted by the application. These other pages will allow for reading terms and conditions or to begin configuring the tool to find their next home. The DNS host will be configured with Amazon's Route53.

There are several ways in which the website may be deployed. For the purposes of Assessor, the web application will be deployed onto Amazon's S3. S3 provides a secure, durable, and highly-scalable cloud-based storage for things like HTML pages, CSS files, images, videos, and JavaScript that make up the application.

When a user is ready to find their home, they will be presented with a form that asks for several features about them. These features, discussed in proceeding model sections, will be serialized to JSON and sent to the Web Server via the server API.

When the model results are returned to the web application, there will be client side processing of each model's results into their corresponding heat-maps, which the user may click between. There will also be an overall heat-map which weighs each model's results and coalesces onto an overall-scored heat-map (shown by default to the user) representing their perfect home given their configuration. The weights, by default, will be equal across all models (i.e. weight = 100% / number of models).

The front-end web-pages will be created using React. React is a modern framework for building interactive web-pages that are fast and portable. React is especially useful because of the ease of creating interactive heat maps, which may be created with react-map-gl for instance [11]. Figure 2 depicts a sample user experience.

Figure 2: Wire Frames of User Experience

2.1.1.2 Web Server

The web server is responsible for processing web requests from the client. It will have 4 API ports to listen onto: /fit, /perception, /financial, and /livability. The server will route each of these requests to their corresponding model via their API, which will be hosted in a private cloud.

When the models have completed processing, results will be sent back to the server, which will then route to the client via a response to respective API.

For the purposes of this component, Amazon's EC2 will be used. EC2 provides computation servers in the cloud, and an easy mechanism for designing the required APIs.

2.1.1.3 Model Architecture

As shown in Figure 1, each model has several components. Each model will be hosted on a virtual machine (EC2 VM), which is responsible for handling requests and processing, as well as returning results back to the server.

When an API call is received, the VM pulls the relevant data from the database (DB), feeds it into the model, runs the model, and feeds the results into the results cache, that is then serialized to JSON and sent to the server via its API.

Important to note is that some models (like the personal fit model) will require DB accesses, while others (such as the sentiment model) may be pre-computed, and thus the request is handled by simply returning the data sitting in the cache. This will vary depending on the implementation of the models; however, it serves as an efficiency in the system processing overheads.

The VM will be hosted on Amazon's EC2, which (as explained in the previous section), is a scalable cloud compute resource. The DB will be hosted on either Amazon RDS (relational database). The cache will be hosted on Amazon S3.

2.1.2 Personal Fit Model

2.1.2.1 Goals

The Personal Fit model aims to answer the following question: "Which neighbourhood does similar individuals reside?" As discussed in Section 1.2, people may look for similar backgrounds or common characteristics in communities. This model takes user's demographic information and attempts to categorize them to similar residents living in a neighbourhood using a classification algorithm known as K-Nearest Neighbours or KNN.

2.1.2.2 Technical Background

One potential classification algorithm is K-Nearest Neighbors, or KNN. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point [12]. KNN is a suitable algorithm for our classification problem for a few reasons. KNN is non-parametric as it does not make any assumptions about the data distribution [13]. This allows it to fit a larger number of functional forms. This is necessary because of the large number of possible neighbourhoods to which the Personal Fit model may classify an individual. Another advantage of KNN is that it doesn't learn a discriminative function from the training data but memorizes the training dataset instead [14]; in other words, it does not require any training.

To implement the KNN algorithm, the following steps will need to be taken [15]:

1. Decide on a similarity or distance metric.
2. Choose an evaluation metric.
3. Choose k, then run k-NN a few times, choosing a different k each time.
4. Optimize k by choosing the one with the best evaluation measure.
5. Test the model by taking user data and classifying.

The distance metric will be the Euclidian distance from the user and the average resident. The evaluation metric that will be implemented is the training and testing on the entire dataset [16]. The model will be tested on the same dataset and evaluated on how well it performed by comparing the predicted response values with the true response values. A precaution needs to be taken against overfitting, and make sure not to over generalize; this will be complemented by the next step of choosing different k values. Finally, the model will be tested by taking new user data that has not yet been labeled and classifying using the model.

2.1.2.3 Implementation

2.1.2.3.1 Data Acquisition

This model is heavily contingent on the available data set. We were originally concerned with what data was readily available and how the data was being presented. A wide variety of data including demographic and income of every neighbourhood in Toronto was found [17]. However, we could not easily find information regarding the relationship between the census tract and hood number or neighbourhood name [18]. This made it difficult to differentiate the census data gathered for different neighbourhoods. As a result, we chose to make a pivot to a more well-organized data set to fit our criteria.

Census data of the population surrounding New York City was found [19]. This was a better choice because it not only had an organized list of all census blocks to its corresponding name, it had more detailed information on its population compared to the census conducted in Toronto. Although it was not feasible to use the Personal Fit model to compare across all profiles gathered in the census, more information meant that we could experiment with which set of information about an individual would best classify them to one of the neighbourhoods.

2.1.2.3.2 Data Persistence

The database containing all the demographic, socio-economical, and selected housing data will be stored [18]. The four excel files included in the zip file shows a very fine grain data for every relation in the schema. During the actual KNN implementation, we will only use a select number of the most discriminating or differentiating factors to perform the analysis. The method of choosing the best subset of relations is through the Analysis of Variance (ANOVA) and the F-test. The F-test is used to determine if the same dimension between two points are statistically significant, that the two points are more correlated [20]. This will be used in our model with the input being a input dimension and the output being the neighbourhood classification. This can be used to determine which subset of the relationship most influences the output.

To remove any redundancies in the data set, the BCNF decomposition should be applied to the relation. All four original excel files provided are already compliant with BCNF, containing no redundancies.

2.1.2.3.3 Technology Stack

To use the KNN algorithm, a software program will be written in Python. Python is chosen over other languages because it offers most functionalities as a more statistical computing language such as R, while being simpler. This makes it easier for us to develop a program in that language. Python provides enough tools to implement KNN with basic libraries. The Python language is also used in the other models in the Assessor tool. This should result in a simpler integration process at the end as well. The program will get the raw data in spreadsheet format from the Amazon S3 cache and RDS database, then pre-compute and validate the KNN algorithm which will be later used to match user's input data to a neighbourhood. Figure 3 shows the full architecture for the Personal Fit model.

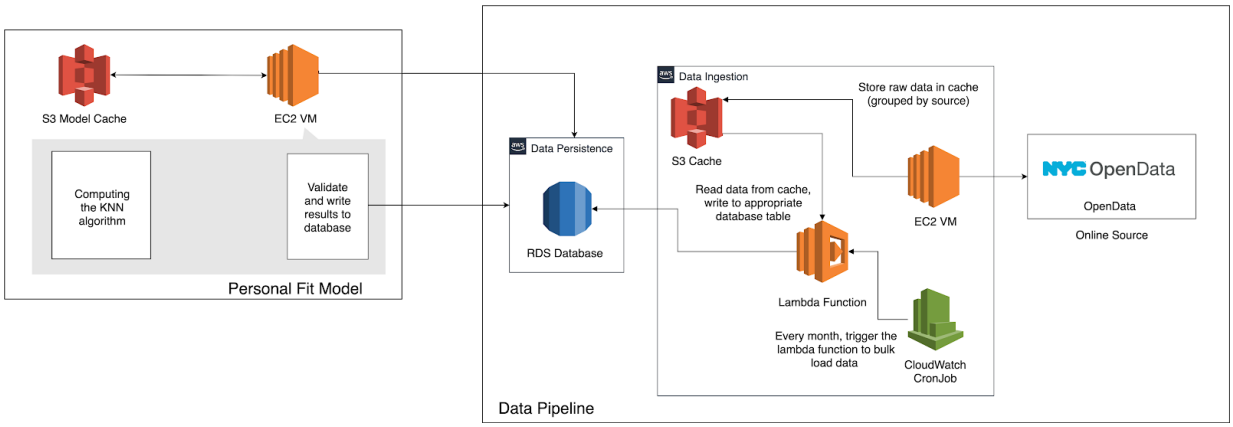


Figure 3: Architectural Diagram for the Personal Fit Model

2.1.2.4 Success Metrics

The Personal Fit model returns the data point that has the closest Euclidean distance with the new point given user input. A holistic metric for success is to express this in a scale of how closely it matches the user, the inverse of the minimum distance will be chosen to have the maximum closeness factor. Using this as a reference, every other relation in the schema will be represented as a percentage (from 0.00 to 1.00).

A technical metric for success will be the accuracy of the KNN algorithm. As mentioned in Section 2.1.2.3.2 the accuracy of the algorithm will depend heavily on which subset of the entire relationship we choose to run the KNN on. This will be completed before training to choose the subset which best segregates the different neighbourhoods with respect to their residents' demographic and socio-economical background. The accuracy will then be determined by comparing the output from the KNN with different subset of the relationships. The farther the distance between K nearest neighbourhood from the user point is, the better.

2.1.2.5 Risks and Mitigation Strategies

Risks associated with the Personal Fit model are found in the algorithm and its design. The KNN algorithm is computationally expensive because the algorithm stores all the training data. This further increases with more relationships or dimensions of user data. This also relates to the design of the algorithm because the KNN suffers from the 'Curse of Dimensionality' [21]. As the number of dimensions increases, the distance between points increases exponentially. This may act in the disinterest of our accuracy success metric of the algorithm; the K nearest points may all be equally far away, thus defeat the purpose of the accuracy metric. These risks will be mitigated by testing for the best subset which produces the best accuracy as described in Section 2.1.2.3.2.

2.1.3 Public Perception Model

2.1.3.1 Goals

The public perception aims to answer the question - "*What does the general public think about a particular neighborhood*?" More specifically, this model crowdsources information from social media and news websites to consolidate an image of how the public views a neighborhood. Often times, home buyers screen neighborhoods based on content they see online, or hear via word of mouth. As discussed in Section 1.2, this rather intangible metric is a critical aspect of the purchase decision. This model replicates this metric via retrieving textual data online, and leveraging various techniques in natural language processing (NLP). The results are quantifiable insights that aid in predicting what neighborhoods are thought of as more favorable by the public.

2.1.3.2 Technical Background

This section discusses high level technical strategies that this model employs. Section 2.1.3.3 further expands the implementation details, and Section 2.1.3.4 details success metrics.

2.1.3.2.1 Sentiment Analysis

One of the biggest challenges of working with textual data is representing large amounts of text in a way easily understood by the end user. Sentiment analysis tackles this problem through establishing a quantitative metric for evaluating text - in particular, the sentiment embedded in the text. Broadly speaking, sentiment analysis looks at whether the text of interest expresses positive or negative sentiments. Some approaches also look at the extent to which a sentence is opinionated as opposed to being objective. Through applying sentiment analysis on a subset of the data relevant to a specific neighborhood, we aim to produce a quantitative value that reflects the public's feelings towards it. Furthermore, we plan to segment the analysis based on data source (e.g., Twitter vs. CNN), to look at how different types of people perceive the neighborhood and analyze the bias underlying each data source.

2.1.3.2.2 Topic Modeling

While sentiment analysis tackles the problem of making sense of large text via quantitative metrics, topic modeling aims to extract textual topics inherent in the text to highlight critical aspects of its meaning. Broadly speaking, these methods seek to summarize large texts by capturing a combination of words that contribute most to its representation. By communicating these topics to the end user of the application, we hope to condense the immense amount of information online into a brief and easily digestible form.

2.1.3.3 Implementation

This section discusses the technical implementation of the public perception model, detailing data acquisition/persistence strategies, and select natural language processing techniques.

2.1.3.3.1 Data Acquisition

To capture a representative sample of the public's perception of a neighborhood, we collect data from both subjective and objective sources. This parallels the purchasing process of a real person, who likely consults with news sources, as well as word of mouth content. For this model, subjective data is sourced from large-scale social media platforms (Twitter and Reddit), and objective data is acquired from reputable news websites (CNN and the New York Times). These sources were selected in part due to ease of data collection - all four of the above specified platforms provide standardized, well documented APIs that will significantly speed up the rate at which a data acquisition engine can be built (detailed below).

Table 1: API Resources for Textual Data Sources

Source	API Documentation	API Endpoint for Search
Twitter	https://developer.twitter.com/en/docs [22]	{root}/search?q={query}
Reddit	https://www.reddit.com/dev/api/ [23]	{root}/r/all/search?q={query}
CNN	http://developer.cnn.com/docs/api/search/ [24]	{root}/newsgraph/search/{key}:{value}
NYTimes	https://developer.nytimes.com [25]	{root}/articlesearch.json?q={query}

The data acquisition engine will utilize these APIs to listen for certain keywords being mentioned on each platform. These keywords will be each neighborhood's name, retrieved from New York City's official Neighborhood Tabulation Area database [26]. For instance, "Greenwich Village" will be searched for. Depending on the coverage retrieved, we may broaden the keyword list to include shortened names of each neighborhood + "NYC". Generally speaking, it is better for us to capture a smaller, more pertinent dataset than it is for us to capture a large, mostly irrelevant dataset (as we want data exclusively on neighborhoods of interest). For instance, we don't want content about Greenwich, UK. Thus, we need to screen for "Greenwich" + "NYC", or "Greenwich" + "New York City".

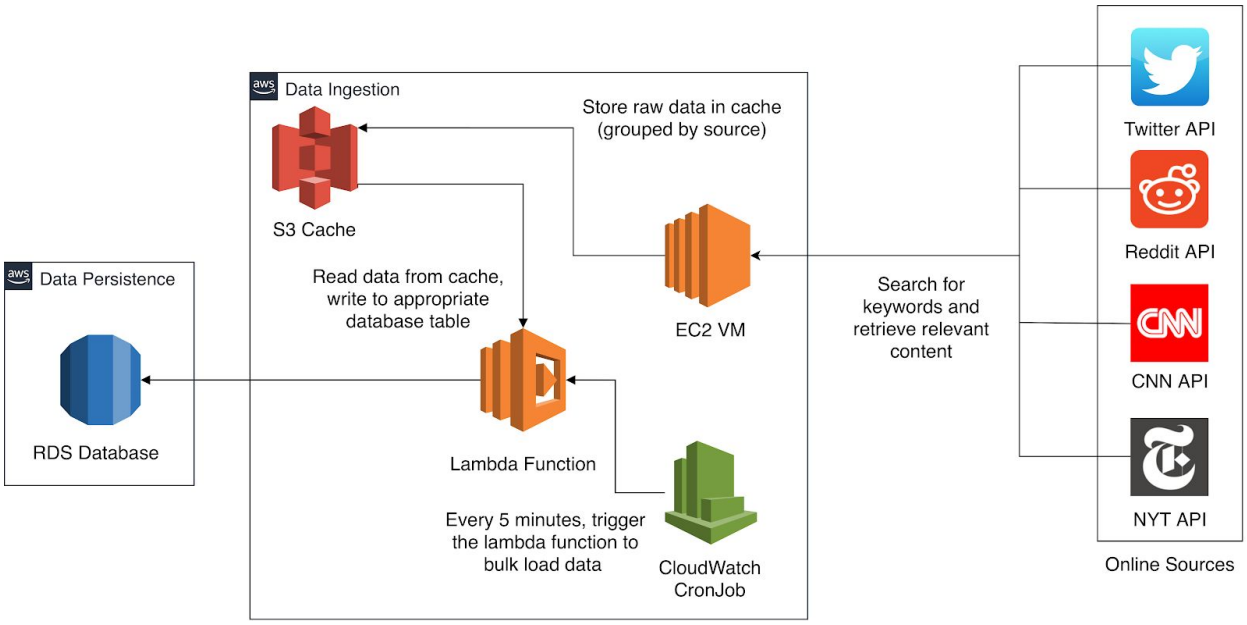


Figure 4: Architectural Diagram for the Data Acquisition Pipeline

Figure 4, above, shows technical details underlying the data acquisition pipeline. Once data is retrieved from online sources via aforementioned APIs by the EC2 Virtual Machine, the raw files (likely in json) will be stored in object storage cache in S3. Then, a CloudWatch CronJob will be leveraged to schedule a task (serverless Lambda) every 5 minutes that loads data from cache, processes it into the desired data persistence format, then stores it into a relational RDS database. Note that this type of workflow enables the data acquisition pipeline to run in real-time. We can always ingest data incrementally from these sources as they come in, and store it in the RDS database for the model to process.

2.1.3.3.2 Data Persistence

The data acquired will be persisted in a relational database (Amazon RDS). We show here the schema for this database, detailing the table format for the data. To derive an efficient representation of the data without redundancies, we used the BCNF decomposition algorithm. We will have five tables to represent our data, all of which are BCNF complaint.

Table 2: Schema for the “IdToSource” Table

Field	Description
id	Unique id for content, assigned by our team (globally unique)
source	Source of the data (e.g.., “Twitter”, “Reddit”, etc.)
source_id	Id for the content, assigned by the source (e.g. tweet_id, reddit_id, etc.)

Table 3: Schema for the “Twitter” Table

Field	Description
id	Id for the tweet assigned by Twitter (unique on Twitter)
user	Username (handle) of the user who posted the content
lang	Language of tweet (e.g., “en” for English)
text	Tweet content
created_at	Datetime in which the tweet was posted
reply_count	Number of replies on the tweet
retweet_count	Number of retweets on the tweet

Table 4: Schema for the “Reddit” Table

Field	Description
id	Id for the post assigned by Reddit (unique on Reddit)
user	Username (handle) of the user who posted the content
lang	Language of reddit post (e.g., “en” for English)
text	Reddit post content
created_at	Datetime in which the post was created
up_count	Number of upvotes on the Reddit post
down_count	Number of downvotes on the Reddit post

Table 5: Schema for the “CNN” Table

Field	Description
id	Id for the article assigned by CNN (unique on CNN)
publish_date	Datetime in which the post was created
headline	Headline of the article
description	Description of the article (high level summary)
s3_content_link	Link to the S3 object containing the full article
url	Url of the news article

Table 6: Schema for the “NYTimes” Table

Field	Description
id	Id for the article assigned by NYTimes (unique on NYTimes)
publish_date	Datetime in which the post was created
headline	Headline of the article
description	Description of the article (high level summary)
s3_content_link	Link to the S3 object containing the full article
url	Url of the news article

A number of design decisions are reflected in the above schema. First, the “IdToSource” table is necessary, to avoid cross-source id duplication - for instance, Twitter and CNN may assign the same id to a tweet and article, respectively. Second, the “reply_count”, “retweet_count” fields on the Twitter database, as well as the “up_count” and “down_count” fields on the Reddit database are collected for potentially interesting analyses. If time permits, these fields will be used to measure the “trustworthiness” of a post - that is, if many users believe a post is pertinent (either via many reposts, or many upvotes), then the post will hold greater sway over the overall evaluation of the neighborhood. Third, a link to S3 static storage object of the text is stored in database, rather than the actual text for news websites. This is due to the fact that news articles are often very long, and when represented as a VARCHAR on the database, will consume a lot of space (adding to costs). By storing a link to cached file instead, we can instead keep track of the raw text separately on S3, where it is much cheaper to maintain.

2.1.3.3.3 Technology Stack

With regards to implementation details of the sentiment analysis model, we will first replicate proven models in literature including the Stanford NLP model [27] and the VADER model [28]. Then, we will undertake development of a bespoke, in-house sentiment model (most likely a convolutional neural network model) trained on publicly available training data (compiled in [29]). After comparing the performance of each model, we plan to attach the most performant one with the final application in production. In case of comparable performance, or competing strengths (e.g., VADER outperforms in-house in subjective data but underperforms in objective data), we will utilize a combined approach that highlights each model’s strength.

With regards to implementation details of the topic modeling model, we plan to leverage both the Latent Dirichlet Allocation (LDA) model [30], as well as the Bi-term Topic Modeling (BTM) model [31]. While both models seek to represent the document as a conglomerate of topics, LDA aims to derive associations between words at a corpus level, while BTM looks at words that frequently appear adjacent to each other. Hence, LDA tends to work better with longer text (in our case, news data), while BTM tends to outperform with short text (in our case, Tweets and Reddit comments). By leveraging their respective strengths, we will compile a list of key topics embedded in each data source, and present it to the end user as such.

Note that prior to running sentiment analysis or the topic modeling, we apply pre-processing to the textual data - including cleaning, filtering, lemmatizing and tokenizing words. These processing steps have been shown numerous times in literature to boost the performance of both tasks [32]. Furthermore, we note that the models will incrementally run on text collected from all four data sources, as the data gets ingested. Once the models are run, the results will be added to another table in the relational database (schema detailed below).

Table 7: Schema for the “NLPData” Table

Field	Description
id	Unique id for content, assigned by our team (globally unique)
stan_sent	Sentiment score assigned by the Stanford NLP model
vader_sent	Sentiment score assigned by the VADER sentiment model
custom_sent	Sentiment score assigned by bespoke, in-house sentiment model
lda_topic_link	Link to the S3 object containing the Latent Dirichlet Allocation topics
btm_topic_link	Link to the S3 object containing the Biterm Topic Modeling topics

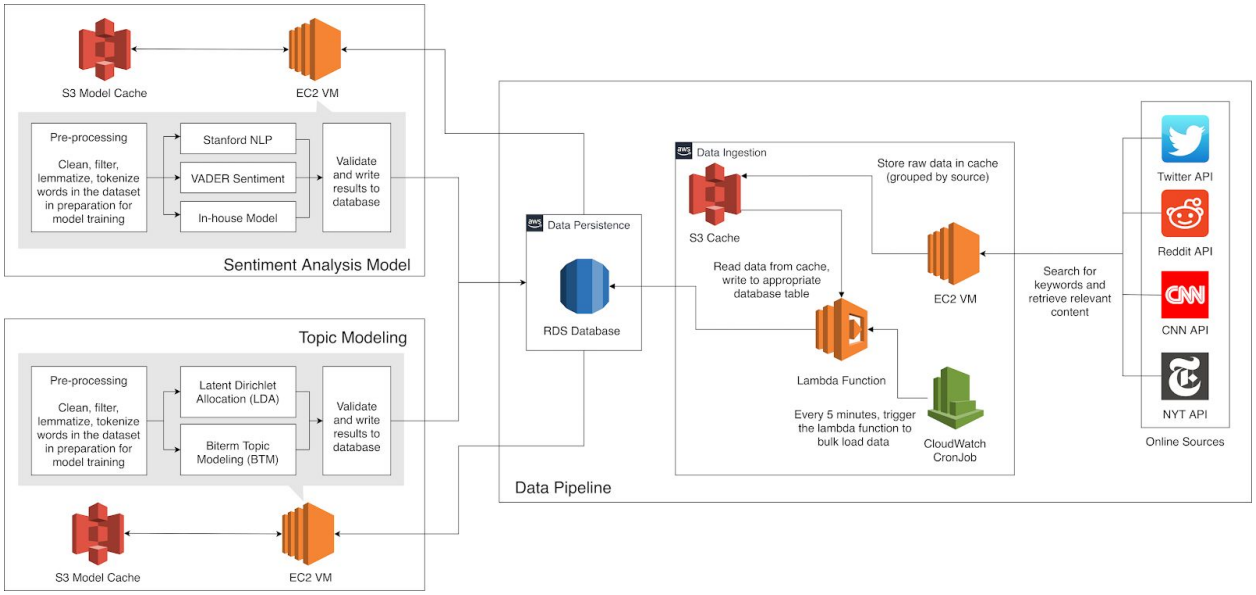


Figure 5: Full Architectural Diagram for the Public Perception Model

2.1.3.4 Success Metrics

This section aims to define metrics through which we can concretely measure the success of our two models. Note that these metrics are derived from literature, and/or previous applications of these techniques in practice.

2.1.3.4.1 Success Metrics for Sentiment Analysis

To evaluate the success of our sentiment analysis algorithm, we will use standardized datasets available online that contain labeled sentences. Prominent ones include labeled tweets (labelled between 0 for the most negative, and 4 for the most positive), as well as labeled movie reviews (labelled between 0 for the most negative, and 10 for the most positive). We first normalize these metrics such that they lie on the same scale. Tentatively, we will assign -2 for the most negative, and 2 for the most positive. Then, the model will be run to evaluate sentiments for a portion of this labeled set. The mean squared error will be evaluated for predicted vs. actual sentiment values.

Time permitting, we plan to extend this rudimentary metric to address the fact that the movie dataset is not domain specific - movie reviews are not necessarily talking about opinions about a neighborhood. While many studies have been conducted that show the legitimacy of using non-domain specific training sets [33] [34], we recognize that this may impact the performance of the model. To address this inconsistency in domain, we plan to mask words that are domain specific. We will first identify words that are movie specific (for movie reviews) and mask them (which in turn, renders them as neutral and not influential for the sentiment evaluation). Thus, only generalizable words will impact the evaluation of text. An example is shown below (actual review from movie dataset).

Original Review
“campbell gives a stunning performance! This twisty plot will have you dazzled for days! Oh yeah, the day a newspaper uses these is the day neve campbell falls in love with me”

Masked Review
“<MASK> gives a stunning <MASK>! This twisty <MASK> will have you dazzled for days! Oh yeah, the day a newspaper uses these is the day <MASK> <MASK> falls in love with me.”

To accomplish this, we use the SpaCy entity recognition library to identify words that are pertinent to movies [35]. Once masking is done, we perform a similar exercise of running the data through our model, and calculating the mean squared error. Note that developing an entity recognition algorithm is a significant task, and it may fall out of scope for our project. Alternatively, we can solve this domain problem by acquiring domain specific training data ourselves. We could use a service like Amazon Mechanical Turk [36] to employ workers that label a portion of the data that we collect ourselves. However, this is rather costly, and is hence out of scope for our project.

2.1.3.4.2 Success Metrics for Topic Modeling

To evaluate the success of our topic modeling algorithm, we will use two metrics widely used in literature. First is the perplexity metric [37], commonly used to evaluate the performance of a probabilistic language model (including topic models). This metric looks at the extent to which the model is generalizable to a test set by evaluating its performance on test data. If the model responds well to new data, its perplexity decreases (entropy is low). This metric is pertinent to our model as it allows us to evaluate the generalizability of our topic model. If our model can condense information for a wide range of data across different sources, it will serve as a valuable resource for individuals seeking to understand public perception. Second is the topic coherence score [38] [39] [40], which measures the semantic coherence of a set of words assigned to a topic. More specifically, it evaluates the pairwise semantic similarity between top N words of a topic, across all topics. Then, it aggregates this measure to produce a holistic metric. This metric is pertinent to our model evaluation as it allows us to quantify the model result's similarity to a human interpretation. If we can condense large texts into small topics that are coherent to humans, it can certainly aid in allowing users to digest the massive amounts of textual data available on the web.

2.1.3.4.2 Success Metrics for Data Pipeline

As the model concerns data being updated in real time (e.g., new posts on Twitter, new articles on CNN), the ability to respond to incoming data in a timely fashion is critical. Thus, we establish a turnaround time goal of 10 minutes for the time that data is captured to when the analysis is available to be rendered on the web application. For instance, if a tweet is captured, its sentiment and topics will be available for the web application within 10 minutes.

2.1.3.5 Risks and Mitigation Strategies

Various risks can be foreseen with regards to the project's completion. First of these concerns the risk surrounding the inability to obtain data. While the four APIs mentioned above are stable and established (and hence carry little risk of going down), they require an application for access. Failure to obtain keys for access will result in inability to listen to data. To mitigate this risk, our team has reviewed the potential reasons for denial of service - which primarily revolves around misuse of their API. Our usage complies with each website's respective terms and services [22] [23] [24] [25], provided that we stay within their call limit (which is well above what we need). To ensure that we stay within this call limit, hard constraints will be set in the software to maintain an appropriate rate of data acquisition.

Another risk surrounds the sheer scale of data that we will be ingesting on a daily basis. Textual data can be quite large, and can quickly consume a lot of storage, ramping up costs. To mitigate this risk, we have designed our database schema to be scalable. More specifically, instead of storing the raw textual data for each document on the relational database, we instead store a link to the textual data file on S3 object storage. This way, we always have access to the raw data corresponding to each document, but do not need to scale our database to extreme sizes. S3 is orders of magnitude less expensive than RDS, and hence, this solution addresses risks surrounding scalability (see Section 2.2 for costs).

The final risk foreseen is the potential of model underperformance. To mitigate this risk, we have conducted extensive literature review (see Section 2.1.3.3.3) to review state-of-the-art work in the field. Through this exercise, we have identified specific techniques that appear to work well in similar contexts. Furthermore, we have selected detailed technical metrics that can monitor the model's performance during development. This includes the mean squared error of sentiment analysis algorithms, as well as the perplexity and topic coherence metrics for the topic model. We will continuously utilize these metrics to identify areas of underperformance, and correct them during model development.

2.1.4 Financial Opportunity Model

2.1.4.1 Goals

The Financial Opportunity model aims to answer the question: are home investments in the neighbourhood favorable? This model employs time series analysis to forecast the future valuations of districts based on historical data. A time series is defined as an ordered sequence of values of a variable at equally spaced time intervals [41]. In the context of this model, the variable is the median sale price of all homes in a neighbourhood. This model forecasts future median sale prices based on historical median sale prices, and provides users with a potential return on their home investments in one to ten years.

2.1.4.2 Technical Background

One approach to time series analysis is to use statistical methods to model the data. The median sale price of all homes in a neighbourhood is an example of a univariate time series. A univariate time series is defined as a time series that consists of single (scalar) observations [43]. Common approaches to modeling univariate time series include triple exponential smoothing and Box-Jenkins approach [44]. The Financial Opportunity model uses the Box-Jenkins approach, an autoregressive moving average (ARMA) model, because similar analysis examples use autoregressive integrated moving average (ARIMA) models [45][46]. The ARIMA model differs from the ARMA model in that non-stationary time series are differenced one or more times to achieve stationarity [47]. The Box-Jenkins approach assumes stationarity, so the model will be some variation of ARMA [47].

The specifications of the Box-Jenkins model are to be determined. There are three stages in building the Financial Opportunity model: identification, estimation, and validation. The identification stage focuses on detecting stationarity and seasonality. A stationary time series is defined as a time series with a mean, variance, and autocorrelation that do not change over time [48]. A seasonal time series is defined as a time series with periodic fluctuations [49]. Stationarity can be determined from a run sequence plot, and seasonality can be determined from an autocorrelation plot [50]. Once stationarity and seasonality are addressed, autocorrelation and partial autocorrelation plots assess the order of the autoregressive and moving average terms [50]. The estimation stage aims to fit the Box-Jenkins model by estimating the parameters. Maximum likelihood estimation is generally the preferred technique [51]. The validation stage attempts to measure the correctness of the chosen model. The Box-Jenkins approach assumes the residuals are white noise with a constant mean and variance [52]. If the assumptions are not satisfied, the process restarts at the identification stage to build a better model.

An alternative approach to time series analysis is to use machine learning methods to model the data. In particular, recurrent neural networks such as long short-term memory (LSTM) neural networks are popular for time series prediction [53]. LSTM is a model organized into cells that passes an internal state variable from one cell to another [53]. The internal state variable is modified by operation gates: forget gate, input gate, and output gate [53]. LSTM models learn which past data is important in making future predictions [53]. The Financial Opportunity model currently intends to employ the Box-Jenkins approach. However, the LSTM model may be considered if the Box-Jenkins approach cannot correctly model the data.

2.1.4.3 Implementation

2.1.4.3.1 Data Acquisition

Zillow provides various historical housing data sets, including their own measures such as Home Index Value and Rent Index, as well as inventory and listing measures such as median list price and median sale price [42]. The Financial Opportunity model is interested in the median sale price by neighborhood, but can be adapted to analyze other data sets. These data sets can be directly downloaded from Zillow in CSV format. The data acquisition engine will download the CSV file, and process the information into raw data that will be sent through the data acquisition pipeline. Figure 5 shows the technical details of the data acquisition pipeline. Similar to the process discussed in the other Assessor models, the cache and database will be updated once a month to add the latest month's data set.

2.1.4.3.2 Data Persistence

A relational database will store the acquired data. The BCNF decomposition algorithm is used to remove redundancies from the relation representing the data. The original relation in the CSV file is: RegionID, RegionName, StateName, SizeRank, and months ranging from Mar-08 to Dec-19; and is already BCNF-compliant.

2.1.4.3.3 Technology Stack

To perform time series analysis on historical median sale prices by neighbourhood, a software program written in Python will model the data. The choice of Python over alternative programming languages with statistical computing capabilities, such as R, is because Python offers much of the same functionalities while also being simpler and consequently quicker to develop in. Both languages provide data analysis tools to implement the Box-Jenkins approach, but Python provides better machine learning tools to implement the LSTM model. Also, a Python program will be consistent with the language used in other Assessor models, thus leading to a more integrated ecosystem in the context of the web architecture. The Python program will get raw data from the Amazon S3 cache and RDS database, model the data to forecast future median sale prices, and write the information back into the cache and database. Figure 6 shows the technical details of the full Future Opportunity model.

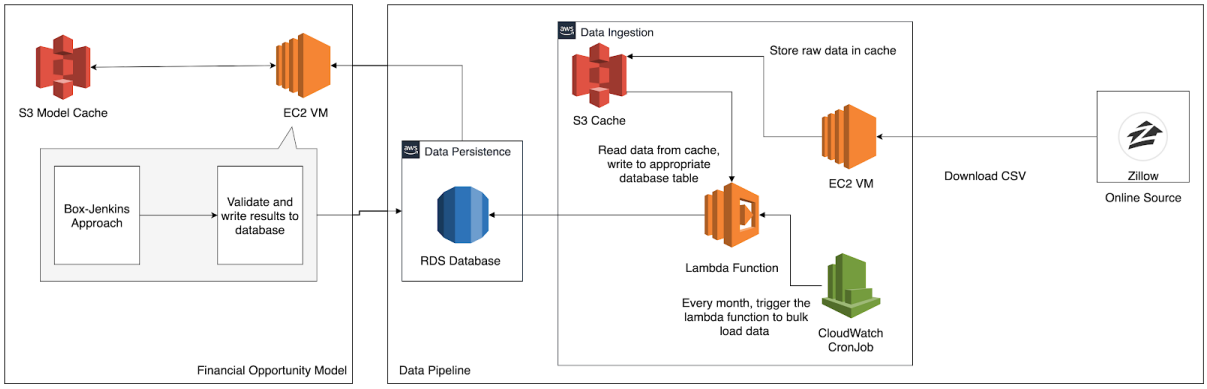


Figure 6: Full Architectural Diagram for the Financial Opportunity Model

2.1.4.4 Success Metrics

To measure the correctness of the Financial Opportunity model, this model will first be built on historical data leading up to one to ten years from the present. The one to ten year forecasts from this model will then be compared against the actual data observed, and a mean square error can be calculated. The performance of this model is not relevant, as the time-consuming analysis is done beforehand, not when a user requests valuation forecasts.

2.1.4.5 Risks and Mitigation Strategies

The main risk of the Financial Opportunity model lies in the Box-Jenkins approach. Specifically, the identification stage of building this model is time-consuming. The identification stage involves detecting and addressing stationarity and seasonality, as well as assessing the order of the autoregressive and moving average terms. All of these steps are a manual process aided by software analysis. Python modules can compute and plot the functions relevant to the characteristics, but the result is determined through manual observation and trial and error. Because each neighbourhood's data set can exhibit different characteristics, a separate model may be built for each district. Repeating the process for each neighbourhood will require too much time. However, the likelihood of each district being unique is low. Rather, there may be a few sets of characteristics that most neighbourhoods belong to. After determining what these characteristics are, the estimation and validation stages are automated.

2.1.5 General Livability Model

2.1.5.1 Goals

The general livability model is intended to identify the quality of life in a particular neighbourhood. It does so by aggregating a set of key indicators about that neighbourhood, and using a baseline for the ideal neighbourhood, classifies the given neighbourhood as having a high quality of life.

2.1.5.2 Technical Background

The General Livability model is based on the Artificial Neural Network (ANN) machine learning model. ANNs are processing models loosely based on the neurological systems found in the mammalian cerebral cortex [54], which help classify a particular pattern as some predetermined object or event, which we will denote as the output set.

ANNs are composed of several layers, each of which consists of several interconnected nodes that have activation functions. In an ANN, patterns are presented via the first layer - the input layer - which communicates to one or more of the subsequent layers - hidden layers - where the bulk of processing is conducted. This processing is comprised of weighted connections that characterize the relevance of particular patterns for the output set. This hidden layer links to the outer layer, where the classification is determined via a probability of belonging to the output set.

For the most part, ANNs have a preset learning rule which adjusts the weights of internal connection in the hidden layer, enabling the model to “learn-by-example”. They are best used in situations for capturing associations or discovering regularities within a set of patterns; where the volume, number of variables or diversity of the data is very great; or, the relationships between inputs are vaguely understood. [54]

Leveraging this model, the General Livability model will help classify a neighbourhood as ‘ideal’ based on a set of relevant inputs, as we describe in the following section.

2.1.5.3 Implementation

Implementing the General Livability model will consist of a pipeline that begins with data collection and preprocessing, and is followed by processing and returning results. We detail the implementation in the following sections.

2.1.5.3.1 Data Acquisition

Determining the ideal district is difficult; however, there are many resources which can help shape the definition with both qualitative factors and quantitative metrics. For the purposes of Assessor, this definition of an ideal district will draw from Mercer. Mercer collects data, annually, about key indicators in cities across the globe to help rank them. Each year they rank over 450 cities worldwide, using 39 key indicators. This data is intended to help consult companies or individuals on the quality of life on foriegn assignments to appropriately price compensation incentives [55].

This model will leverage the key indicator classes collected by Mercer (29 of which were explicitly stated and the others remain undisclosed, with relevant indicators summarized in Table 8) for each city currently ranked on the Mercer ranking to learn what makes a city “ideal”. These classes of indicators will then be mined from the internet per district per city and fed into the model to help determine the districts classification as an ideal neighbourhood to live in. For the scope of this project, the data will be pulled from New York Open Data [56] and Neighbourhood Scout [57]

Figure 7 depicts the data acquisition pipeline. The frequency of data collection will be on the order of 12 months, as these particular district features do not change as frequently as those in the public perception model.

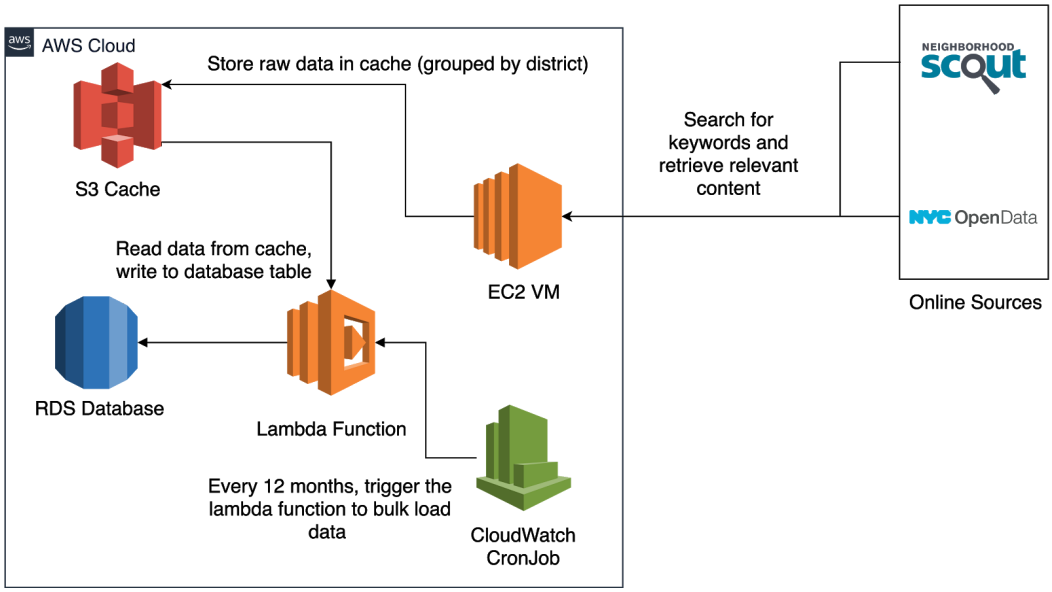


Figure 7: Architectural Diagram for the Data Acquisition Pipeline

Table 8: Key Indicators Used to Assess a City

Category	Indicator
Political and social environment	Political stability
	Crime
	Law enforcement
Economic environment	Banking services
Socio-cultural environment	Media availability and censorship
Medical and health considerations	Medical supplies and services
	Infectious diseases
	Sewage
	Waste disposal
	Air pollution
Schools and education	Standards and availability of schools
Public services and transportation	Electricity
	Water
	Public transportation
	traffic congestion
Recreation	Restaurants
	Theatres and Cinemas
	Sports and leisure
Consumer goods	Availability of food / daily consumption items
	Cars
Housing	Rental housing
	Maintenance services
Natural environment	Record of natural disasters
	Climate

2.1.5.3.2 Data Persistence

Once the data has been collected, it will be processed so as to be readily ingested by the ANN. To ensure the model is fed the appropriate data, the relational schema shown in Figure 8 will be used, which is BCNF compliant [58].

```
Relation (
        district_id,
        political_stability_rate,
        crime_rate,
        police_per_resident,
        banking_services_per_resident,
        censorship_rate,
        medical_services_per_resident,
        diseases_rate,
        sewage_per_resident,
        waste_disposal_rate,
        air_pollution_density,
        schools_per_resident,
        electricity_availability,
        water_cleanliness_rating,
        public_transport_stops_per_resident,
        traffic_congestion_rate,
        restaurants_per_resident,
        theatres_per_resident,
        sports_leisure_per_resident,
        consumer_goods_availability,
        cars_per_resident,
        rental_housing_per_resident,
        maintenance_services_per_resident,
        num_natural_disasters,
        climate_class
)
```

Figure 8: Table Schema for General Livability Model

2.1.5.3.3 Technology Stack

To implement the General Livability model, the python programming language will be leveraged. Python has a lot of support for machine learning applications. In particular, there are numerous libraries which have optimized many machine learning toolkits. For the artificial neural net instance, the sklearn framework may be leveraged, using the Multi-layer Perception module (MLPClassifier) [59]. At the writing of this report, the exact framework has not been decided upon, as this model is out of scope.

In Figure 9 we depict the model architecture in detail. Note the usage of the Amazon AWS stack as detailed in Section 2.1.1.

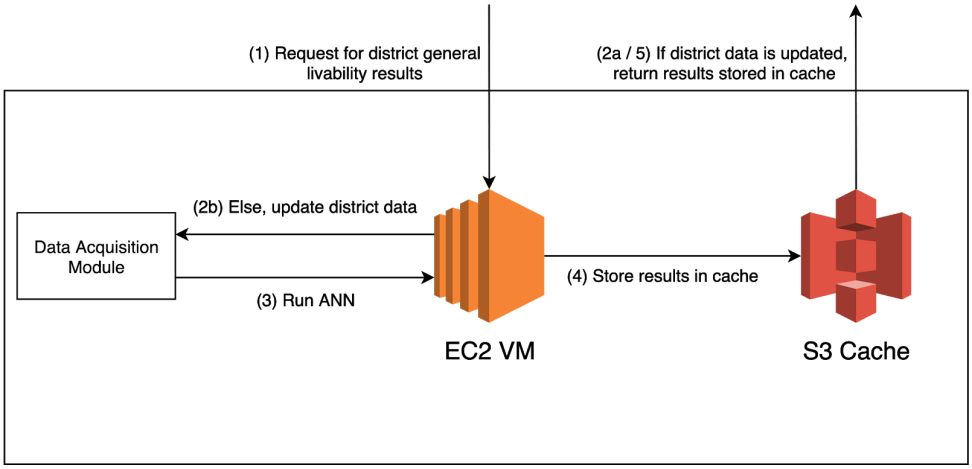


Figure 9: Architectural Diagram for the General Livability Model

2.1.5.4 Success Metrics

Evaluating the model's success from a technical standpoint may be achieved by selectively training on different sets of data. To train the model, we will leverage Mercer results from the previous 5 years. We will feed the model the characteristics of select cities on the mercer list, and ensure it learns to correctly score cities in a manner which reflects the mercer ranking for that particular year. For example, if Toronto scored above New York, we will want to see the score that the model generates reflect this fact; we will do this for multiple (100+) cities. What this does is helps the model learn the characteristics of good cities, which it may use to classify any place given its key indicators. We will then leverage the reserved testing set for this year to assess the model's accuracy. Note that a city and a district will be indistinguishable, as key indicators are not unique to either, thus we may train our model on cities but test it on districts.

Once the model achieves a desirable error rate, it will be fed key indicators about each district of a city, and return a score which will indicate that particular districts probability of being the best place to live based on quality of life. This will help answer the intended question for this model, and add an additional layer of confidence that Assessor's recommendation is accurate.

2.1.5.5 Risks and Mitigation Strategies

As this model depends on the Mercer list, gathering the training and validation data will be heavily reliant on Mercer. There are two areas of risk with this: the first is that Mercer may be hard to reach; and the second is that the data may be expensive. In the event that Mecer is hard to contact, we will ensure we contact early in the semester, and leverage professional networks to help guide us to the right people to contact. If we succeed in contact and they are unwilling to work with us, we will mine the data on our own - which has its own set of risks (described below). If the cost associated with obtaining the data is too high, we will, again, mine the data on our own.

Mining the required data presents a set of risks which may be detrimental to the model's success. The first risk is associated with time required. As the internet is so large and data mining a complex task - with many webpages stopping mining bots for instance - the time required to gather the data for training, validation, and testing, as well as for each district within the city may extend beyond the time frame of the course. Another risk is associated with getting the right data (down to neighbourhood granularity). If either of these risks go unaddressed, the model will be incomplete. To address these issues, we will ensure we are leveraging datasets which are open to the public, and that we scope to a city which has such datasets. If either of these is not possible, we will seek alternative data sources which rank cities, such as bestcities.org's [60] or the Global Liveability Rankings (highly similar to Mercer) [61].

A risk with the model itself is lack of accuracy. As discussed in 2.1.5.4, the model may be trained on cities but used to classify neighbourhoods. One could argue that this difference may reduce model accuracy; however, the data (key indicators) are normalized (by rate or per person for instance). This ensures the model sees only characteristics - irrespective of whether they come from a country, city, or neighbourhood; although, the model will be trained on data that closely resembles neighbourhood granularity.

2.1.6 Combining Results

Each model returns a result which is unique to the model. The personal fit model returns a probability, the public perception model returns a score, the financial model returns a value, and the general livability model returns a probability. To combine results and inform the user of the best neighbourhood, we normalize results.

Since the Personal Fit and General Livability models return probabilities they do not need to be normalized. The public perception model may be normalized by taking the maximum mean

sentiment value over all neighborhoods, and basing all other sentiment values against this sentiment value (i.e. result = value / max_value). Similarly, the financial model may be normalized by taking the maximum appreciating value of all neighbourhoods, and basing all appreciating values against this maximum value.

When all results are normalized, we then apply the default weights to each (100% / number of models, which the user may configure thereafter) and present neighbourhoods according to the highest score.

Following the recommendations, we will assess Assessors effectiveness by surveying trial users. This will enable us to evaluate the holistic performance of product, and help us modify it where required. This survey is in scope; however it may take the form of a pen-and-paper assessment or a web-based feedback form; time permitting.

2.2 Costs

Since Assessor is built from scratch leveraging open-source libraries, the primary costs will be incurred by the hosting of the application on servers.

Amazon's software stack provides a flexible fee structure which makes hosting the application economical. The S3 instance costs, at a maximum, about US$0.03 / Month [62]; while the EC2 instance costs US$0.006 to US$7.00 / hour [63], depending on how powerful the computation server is required to be. For the databases, RDS will cost anywhere from US$0.04 to US$9.00 / hour (heavy storage requirements) [64]. The website domain name was CA$13. Table 9 summarizes costs.

Table 9: Cost Breakdown in Implementing Assessor

Description	Unit Cost (US$)	Total Cost (US$)
S3	0.03 @ 2 months	0.06
EC2	1 @ 100 hours	100
RDS	xlarge 0.29 @ 2 months @ 24hrs	390
Domain name	9.90	9.90
Net Costs		500

An important point to note is that development need not take place online - thus costs will only be incurred once Assessor is ready to be deployed to production. We see that the RDS instance is quite expensive; however, we estimated an xlarge for two months which is conservative. With this fact, it is clear that the costs of hosting the application on Amazon's stack are within reason and in-scope.

2.3 Legality

For each model, Assessor will be leveraging publically available datasets; in particular, news stories, social media posts, and census data. To obtain these datasets, we will be making use of APIs that are built for exactly this purpose - and as discussed previously, our use will comply with the terms and conditions outlined by respective APIs. Data collected from News APIs pertain to already public content, and hence are not causes for concern. For data collected from social media, the respective APIs already mask out personally identifiable information. For the personal fit model, data about individuals will be collected; however, this information is not personally identifiable (no names, emails, or addresses), and this data will not be stored but only processed by the model to obtain results, thus protecting the individual's privacy.

2.4 Task Delegation and Project Management

Given the many components involved in creating the Assessor tool, it's development has been broken into a set of milestones, each of which have been designated to each team member (responsible).

2.4.1 Responsibles

In Table 10a and Table 10b we present the task delegation amongst group members. Though each milestone has an assigned responsible, members will provide assistance where required.

2.4.1.1 Eric Boszin

Eric will be responsible for orchestrating the system architecture. In particular, he will be developing the front-end, back-end server APIs, and deploying the application to the cloud. He will also be responsible for assisting members in model development - and time permitting - building and deploying the General Livability model. Eric has experience in project management and developing web-based applications, which will help him manage team deliverables.

2.4.1.2 Henry Cho

Henry will be leading the development of the Public Perception model, leveraging techniques in natural language processing including sentiment analysis and topic modeling. Alongside this, he will be responsible for building a data pipeline for the model, enabling real-time ingestion of online data. He will also work with Eric in all things cloud related, particularly on affairs related to application deployment, and collaborate with Dale on a need-basis. Henry has experience in data engineering and data science, which will aid in this line of work.

2.4.1.3 Dale Wu

Dale will be taking ownership of the Financial Opportunity model using the Box-Jenkins approach. He will collect housing data from Zillow, and build distinct models for each district. The models will be validated and tested against observed historical data. Dale will also integrate his completed models into the VM on the cloud. In addition, he will be working with Henry on the Public Perception model as necessary.

2.4.1.4 Shinjae Yoo

Shinjae will be mainly responsible for implementing the Personal Fit model using the KNN algorithm. He will architecture and validate the test model, then, move the model to the cloud and integrate with the VM that runs the Assessor tool. Shinjae has experience coding with a big project and theoretical background necessary for the model implementation.

Task Name	Assigned To
Complete project proposal	All
Complete initial feasibility report	All
Complete final feasibility report	All
Develop skeleton front end (local)	Eric
Design architecture for the eventual deployment onto the cloud	Eric
Develop schema for the API used by application to access data	Eric
Hook up skeleton with API serving dummy data	Eric
Polish and complete skeleton front end development	Eric
Deploy front end resources onto the cloud and make it publicly available	Eric
Develop back end API for fetching data from models and serving to application	Eric
Implement logging and caching for easier access to frequently requested data	Eric
Integrate back end API with the front end application, replacing dummy API	Eric
Implement rigorous tests on availability, functionality and edge cases	Eric
Implement heat maps to better represent the data to end users on the application	Eric
Implement visualizations to help users understand why regions were recommended	Henry
Render explainable AI visualizations (e.g. LIME) to increase faith in model	Eric
Move the stack over to serverless technologies, leveraging Kubernetes.	Eric
Implement alerting services for developers to be made aware of downtime.	Eric

Table 10a: Responsible Breakdown for System Architecture

Task Name	Assigned To
Research and design architecture for the personal fit model	Shinjae
Research and consolidate census data required for the model	Shinjae
Implement and train classification model with harvested data	Shinjae
Validate and test model with human interpretable test cases	Shinjae
Move the model to the cloud - launch and set-up a VM that runs the model	Shinjae
Wrap the model around an API queryable by the application's back end API	Shinjae
Implement caching and logging for the model	Shinjae
Research and design architecture for the financial opportunity model	Dale
Research and consolidate housing prices data required for the model	Dale
Implement and train statistics based time series model	Dale
Implement and train recurrent neural network / LSTM based time series model	Dale
Back test the predictions made by the model on historical data	Dale
Wrap the model around an API queryable by the application's back end API	Dale
Implement caching and logging for the model	Dale
Research and design architecture for the public perception model	Henry
Develop data pipeline for harvesting data from Twitter, Reddit and News sources	Henry
Develop sentiment models, leveraging Stanford NLP, Vader etc.	Henry
Develop topic modeling techniques, allowing extraction of topics from sentences	Henry
Validate and test model with human interpretable test cases	Henry
Wrap the model around an API queryable by the application's back end API	Henry
Implement caching and logging for the model	Henry

Develop rigorous test cases for integration of models with the application, and debug	All
Research and design architecture for the general liveability model	Eric
Consolidate data required for the model, including crime data, various indices etc.	Eric
Develop neural networks that enable prediction based on quantitative data	Eric
Develop human interpretable test cases for the liveability model	Eric
Wrap the model around an API queryable by the application's back end API	Eric
Implement caching and logging for the model	Eric
Set up alerting services on host virtual machines to ensure developers are aware of bugs	All

Table 10b: Responsible Breakdown for Model Development

2.4.2 Agile Software Development

To aid in achieving project milestones, the team is drawing from the agile software development framework. Agile software development seeks to deliver working software in frequent increments, and welcomes changing requirements as the product evolves. It has been proven to work effectively by many organizations engaged in the development of large scale software applications.

The development of the Assessor tool will be broken down into weekly sprints, where a particular milestone is expected to be completed and functioning. This breakdown is presented in the following section, section 2.5. To gauge the progress of Assessor's development, number of working software components and ability to adhere to sprint timelines will be used as a metric for project progress.

To aid in collaboration, we will employ google colab - a useful tool for developing software (particularly python) on the cloud.

2.5 Roadmap

In this section, Gantt charts are leveraged to determine an appropriate scope for the project given the time constraints. In the charts, broad technical segments of the project are broken down into granular subtasks, each with a planned window for completion. Due to the large number of subtasks, the Gantt chart is divided into two. The first outlines tasks related to web/cloud architecture and general administrative tasks. The second outlines tasks related to the development of the four models (personal fit, public perception, financial opportunity, and general liveability). Note that the chart extends beyond the time frame allotted for the project.

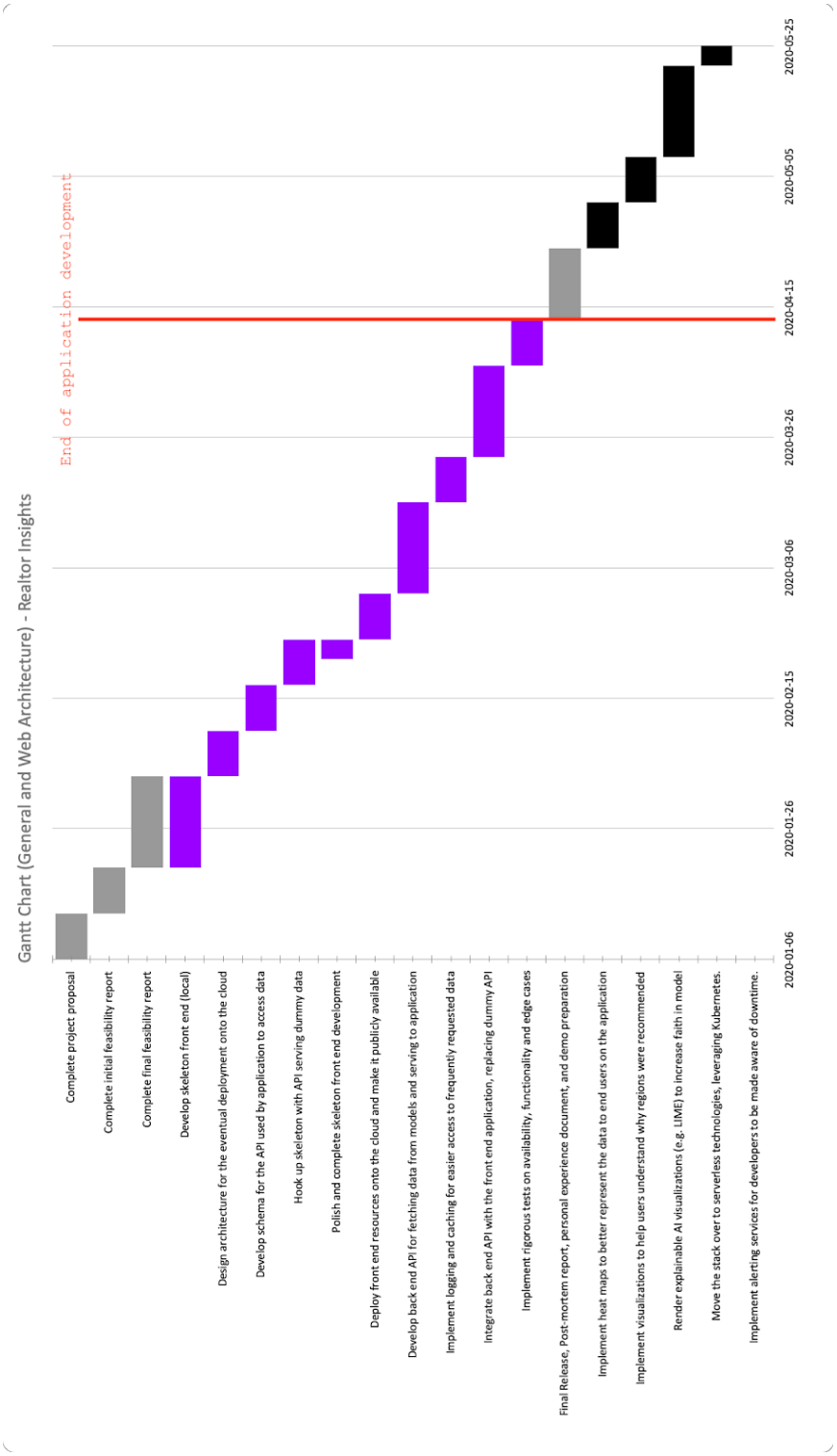


Figure 10a: Sprint Breakdown Concerning Administrative in Gray and System Architecture in Purple. Black Indicates Out-of-Scope.

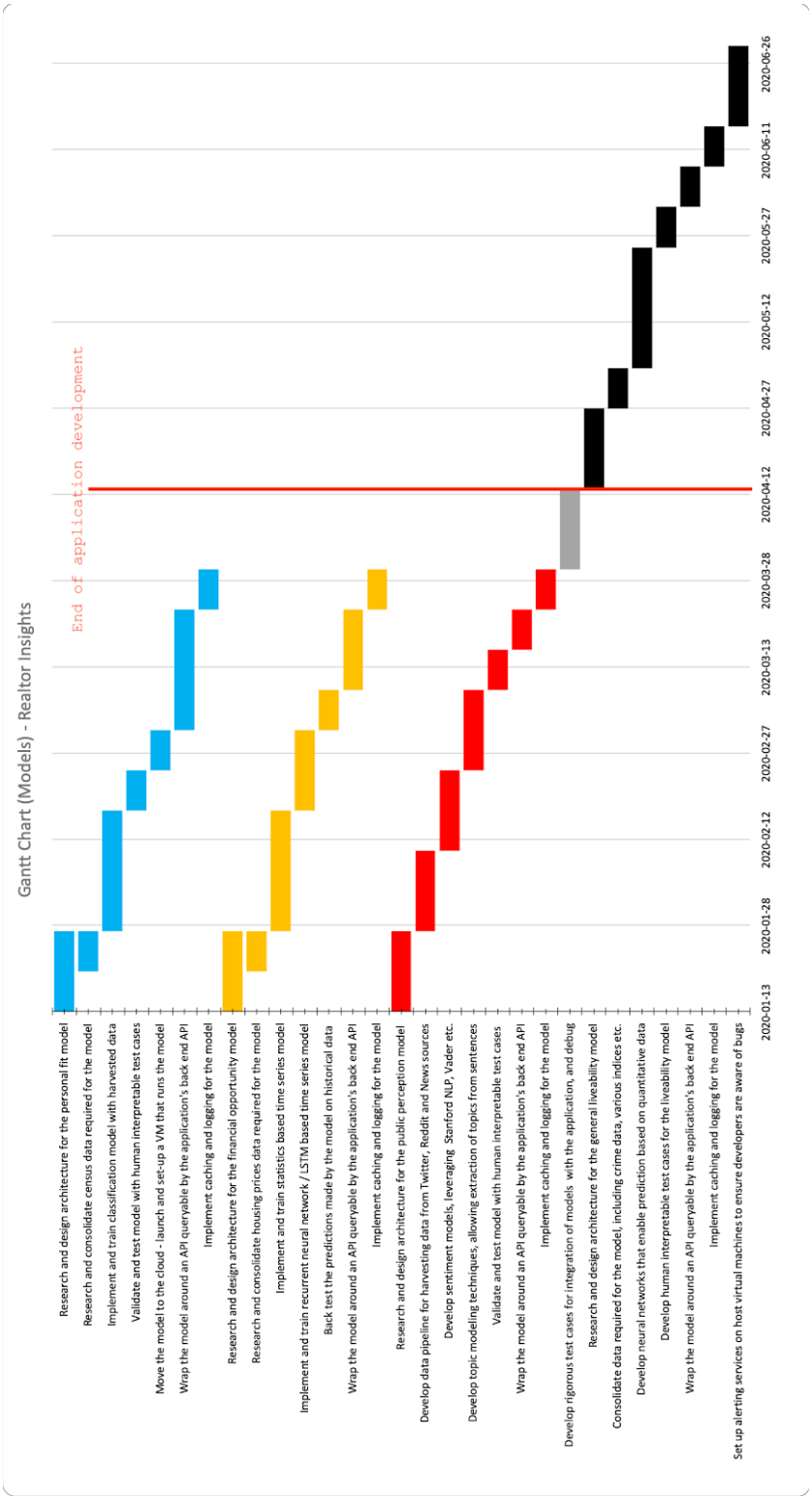


Figure 10b: Sprint Timeline Concerning Personal Fit Model in Blue, Financial Opportunity Model in Yellow, Public Perception Model in Red, and All Models in Grey. The General Liveability Model is Out-of-Scope.

3 Recommendations

Assessor is a promising tool which may help individuals looking for their next home make a more informed decision. Given that this is a Proof of Concept, and the constraints on timelines for the course, we conclude that the General Livability Model, alerting services, and the model visualization and heat maps are out of scope (Figure 10a and Figure 10b). Instead, a user will be presented with a basic table with the results per model, per neighbourhood. If, however, we are able to reach milestones earlier than anticipated, we may pursue the aforementioned features.

4 References

[1] “Move to your best place to live and work,” Teleport Cities. [Online]. Available: https://teleport.org/cities/. [Accessed: 21-Jan-2020].

[2] N. Mansur, “Property Finder Tool - Finding Real Estate Investment Properties in 2018: Mashvisor,” Investment Property Tips | Mashvisor Real Estate Blog, 25-Feb-2019. [Online]. Available: https://www.mashvisor.com/blog/property-finder-tool-mashvisor/. [Accessed: 21-Jan-2020].

[3] V. Daibes, “How to Find the Best Investment Property Using a Heatmap: Mashvisor,” Investment Property Tips | Mashvisor Real Estate Blog, 05-Feb-2019. [Online]. Available: https://www.mashvisor.com/blog/find-the-best-investment-property-using-a-heatmap/. [Accessed: 21-Jan-2020].

[4] B. Martucci, “11 Tips to Find the Best Neighborhood to Live In,” Money Crashers, 26-Dec-2018. [Online]. Available: https://www.moneycrashers.com/tips-find-best-neighborhood-live-in/. [Accessed: 03-Feb-2020].

[5] L. Gray, “How to Choose a Neighborhood,” HGTV. [Online]. Available: https://www.hgtv.com/design/real-estate/how-to-choose-a-neighborhood. [Accessed: 03-Feb-2020].

[6] D. Schmidt, “How to Pick and Move to the Best Neighborhood for You and Your Family,” The Spruce, 25-Sep-2019. [Online]. Available: https://www.thespruce.com/choosing-the-right-neighborhood-2435878. [Accessed: 03-Feb-2020].

[7] “Vienna Tops Mercer's 21st Quality of Living Ranking,” Mercer. [Online]. Available: https://www.mercer.com/newsroom/2019-quality-of-living-survey.html. [Accessed: 03-Feb-2020].

[8] P. M. Heathcote and S. Langfield, A level computing. Ipswich: Payne-Gallway, 2004.

[9] “S3,” Amazon, 2002. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 21-Jan-2020].

[10] D. J. Daly and D. J. Daly, “Economics 2: EC2,” Amazon, 1987. [Online]. Available: https://aws.amazon.com/ec2/. [Accessed: 21-Jan-2020].

[11] “React components for Mapbox GL JS,” react-map-gl. [Online]. Available: https://uber.github.io/react-map-gl/#/Documentation/introduction/introduction. [Accessed: 21-Jan-2020].

[12] A. Bronshtein, “A Quick Introduction to K-Nearest Neighbors Algorithm,” Medium, 06-May-2019. [Online]. Available: https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7. [Accessed: 21-Jan-2020].

[13] J. Brownlee, “Parametric and Nonparametric Machine Learning Algorithms,” Machine Learning Mastery, 24-Oct-2019. [Online]. Available: https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/. [Accessed: 21-Jan-2020].

[14] “Why is Nearest Neighbor a Lazy Algorithm?,” Dr. Sebastian Raschka, 20-Jan-2020. [Online]. Available: https://sebastianraschka.com/faq/docs/lazy-knn.html. [Accessed: 21-Jan-2020].
[15] D. Haroon, “How is the k-nearest neighbor algorithm different from k-means clustering?,” Quora, 02-Nov-2018. [Online]. Available: https://www.quora.com/How-is-the-k-nearest-neighbor-algorithm-different-from-k-means-clustering. [Accessed: 21-Jan-2020].
[16] “K-nearest Neighbors (KNN) Classification Model,” ritchieng.github.io. [Online]. Available: https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/. [Accessed: 21-Jan-2020].
[17] City of Toronto, “Neighbourhood Profiles,” City of Toronto, 05-Dec-2018. [Online]. Available: https://www.toronto.ca/city-government/data-research-maps/neighbourhoods-communities/neighbourhood-profiles/?fbclid=IwAR3XBlB1X07fHAQMvTCYhdzsbajypqxrJIpW-m-zCq-q8elW13G7qLStcX4. [Accessed: 21-Jan-2020].
[18] City of Toronto, “Census Tract Reference Maps,” City of Toronto, 15-Nov-2017. [Online]. Available: https://www.toronto.ca/city-government/data-research-maps/maps/census-tract-reference-maps/?fbclid=IwAR00Ei1tlFe5iSm4GSd8WUee_3uzGrAI9A9ZGR6wC5srL9WK2CY4RBoMth4. [Accessed: 21-Jan-2020].
[19] D. of C. P. (DCP), “Demographics and profiles at the Neighborhood Tabulation Area (NTA) level: NYC Open Data,” Demographics and profiles at the Neighborhood Tabulation Area (NTA) level | NYC Open Data, 17-Feb-2015. [Online]. Available: https://data.cityofnewyork.us/City-Government/Demographics-and-profiles-at-the-Neighborhood-Tabu/hyuz-tij8?fbclid=IwAR24-fawwhoLS_meww5YIrQoEbuVE23vsqqWIB5171dKmHdBt3Hm0LF1734. [Accessed: 21-Jan-2020].
[20] “F Statistic / F Value: Definition and How to Run an F-Test,” Statistics How To. [Online]. Available: https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/f-statistic-value-test/. [Accessed: 03-Feb-2020].
[21] P. Grant, “k-Nearest Neighbors and the Curse of Dimensionality,” Medium, 24-Jul-2019. [Online]. Available: https://towardsdatascience.com/k-nearest-neighbors-and-the-curse-of-dimensionality-e39d10a6105d. [Accessed: 03-Feb-2020].
[22] "Docs", *Developer.twitter.com*, 2020. [Online]. Available: https://developer.twitter.com/en/docs. [Accessed: 04- Feb- 2020].
[23] "reddit.com: api documentation", *Reddit.com*, 2020. [Online]. Available: https://www.reddit.com/dev/api/. [Accessed: 04- Feb- 2020].
[24] "NewsGraph Search | CNN", *Developer.cnn.com*, 2020. [Online]. Available: http://developer.cnn.com/docs/api/search/. [Accessed: 04- Feb- 2020].

[25] *Developer.nytimes.com*, 2020. [Online]. Available: https://developer.nytimes.com. [Accessed: 04- Feb- 2020].
[26] *Data.cityofnewyork.us*, 2020. [Online]. Available: https://data.cityofnewyork.us/City-Government/Neighborhood-Tabulation-Areas-NTA-/cpf4-rkhq. [Accessed: 04- Feb- 2020].
[27] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. [Online]. Available: http://nlp.stanford.edu:8080/sentiment/rntnDemo.html. [Accessed: 21-Jan-2020].
[28] C. J. Hutto and E. Gilbert, “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text,” International Conference on Weblogs and Social Media, May 2014.
[29] Datasets. [Online]. Available: http://www.cs.cornell.edu/home/llee/data/. [Accessed: 21-Jan-2020].
[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” Journal of Machine Learning Research 3, vol. 3, Mar. 2003.
[31] X. Cheng, X. Yan, Y. Lan and J. Guo, "BTM: Topic Modeling over Short Texts," in IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 12, pp. 2928-2941, 1 Dec. 2014. doi: 10.1109/TKDE.2014.2313872
[32] M. Nayebi, H. Cho and G. Ruhe, “App Store Mining is not enough for App Development”, in Empirical Software Engineering Journal, 2018.
[33] B. Liu, “Sentiment Analysis and Opinion Mining”, in Synthesis Lectures on Human Language Technologies, vol 5, no. 1, pp. 1-167, May 2012.
[34] K. Denecke, “Using SentiWordNet for Multilingual Sentiment Analysis”, in IEEE Conference on Data Engineering, 7 Apr. 2008.
[35] *Spacy.io*, 2020. [Online]. Available: https://spacy.io. [Accessed: 04- Feb- 2020].
[36] "Amazon Mechanical Turk", *Mturk.com*, 2020. [Online]. Available: https://www.mturk.com. [Accessed: 04- Feb- 2020].
[37] B. Soltoff, "Topic modeling | Computing for the Social Sciences", *Computing for the Social Sciences*, 2020. [Online]. Available: https://cfss.uchicago.edu/notes/topic-modeling/. [Accessed: 04- Feb- 2020].
[38] D. Newman, J. H. Lau, K. Grieser and T. Baldwin, “Automatic Evaluation of Topic Coherence” in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computation Linguistics, pp. 100-108, Jun. 2010.
[39] J. H. Lau, D. Newman, T. Baldwin, “Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality”, in Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pp. 530-539, 2014.
[40] K. Stevens, P. Kegelmeyer, D. Andrzejewski, D. Buttler, “Topic Coherence over many models and many topics”, in Empirical Methods in Natural Language Processing, Jul. 2012.
[41] *6.4.1. Definitions, Applications and Techniques*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm. [Accessed: 21-Jan-2020].

[42] “Housing Data,” *Zillow Research*. [Online]. Available: https://www.zillow.com/research/data/. [Accessed: 21-Jan-2020].
[43] *6.4.4. Univariate Time Series Models*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc44.htm. [Accessed: 21-Jan-2020].
[44] *6.4.4.4. Common Approaches to Univariate Time Series*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc444.htm. [Accessed: 21-Jan-2020].
[45] F. Aguilar, “Time Series Analysis on US Housing Data,” *Medium*, 15-Jul-2019. [Online]. Available: https://medium.com/@feraguilari/time-series-analysis-modfinalproyect-b9fb23c28309. [Accessed: 21-Jan-2020].
[46] B. Dhar, “Time Series Analysis of House Price Index of East South Central Division,” 09-Dec-2017. [Online]. Available: http://rstudio-pubs-static.s3.amazonaws.com/339483_08ba8b928339464e916e54901156b348.html. [Accessed: 21-Jan-2020].
[47] *6.4.4.5. Box-Jenkins Models*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc445.htm. [Accessed: 21-Jan-2020].
[48] *6.4.4.2. Stationarity*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm. [Accessed: 21-Jan-2020].
[49] *6.4.4.3. Seasonality*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc443.htm. [Accessed: 21-Jan-2020].
[50] *6.4.4.6. Box-Jenkins Model Identification*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc446.htm. [Accessed: 21-Jan-2020].
[51] *6.4.4.7. Box-Jenkins Model Estimation*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc447.htm. [Accessed: 21-Jan-2020].
[52] *6.4.4.8. Box-Jenkins Model Diagnostics*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc448.htm. [Accessed: 21-Jan-2020].
[53] N. Pant, “A Guide For Time Series Prediction Using Recurrent Neural Networks (LSTMs),” *Medium*, 07-Mar-2019. [Online]. Available: https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f. [Accessed: 21-Jan-2020].
[54] Pages.cs.wisc.edu. (n.d.). A Basic Introduction To Neural Networks. [online] Available at: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html [Accessed Feb. 2020].
[55] “Solutions,” Quality of Living Data and Hardship Premiums for International Assignments. [Online]. Available: https://mobilityexchange.mercer.com/quality-of-living. [Accessed: 21-Jan-2020].
[56] City of New York, N. (n.d.). NYC Open Data. [online] Opendata.cityofnewyork.us. Available at: https://opendata.cityofnewyork.us [Accessed Feb. 2020].
[57] Cs.toronto.edu. (n.d.). Boyce–Codd Normal Form (BCNF). [online] Available at: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk12/wk12_BCNF2-up.pdf [Accessed Feb. 2020].

[58] Neighborhoodscout.com. (n.d.). New York Crime Rates and Statistics - NeighborhoodScout. [online] Available at: https://www.neighborhoodscout.com/ny/new-york/crime [Accessed Feb. 2020].
[59] Scikit-learn.org. (n.d.). 1.17. Neural network models (supervised) — scikit-learn 0.22.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/neural_networks_supervised.html [Accessed Feb. 2020].
[60] Best Cities. (n.d.). The World's Best Cities. [online] Available at: https://www.bestcities.org/rankings/worlds-best-cities/ [Accessed Feb. 2020].
[61] Eiu.com. (n.d.). Global Liveability Ranking. [online] Available at: http://www.eiu.com/topic/liveability?zid=liveability2019&utm_source=economist-daily-chart&utm_medium=link&utm_name=liveability2019 [Accessed Feb. 2020].
[62] “S3,” Amazon, 2002. [Online]. Available: https://aws.amazon.com/s3/pricing/. [Accessed: 21-Jan-2020].
[63] D. J. Daly and D. J. Daly, “Economics 2: EC2,” Amazon, 1987. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/. [Accessed: 21-Jan-2020].
[64] “RDS: understanding animal research in medicine,” Amazon, 2007. [Online]. Available: https://aws.amazon.com/rds/postgresql/pricing/. [Accessed: 21-Jan-2020].