

# 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 (with the exception of 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 these solutions are promising, they fail when it comes to finding the perfect home; it is more for those interested in rental properties.



## 1.2 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

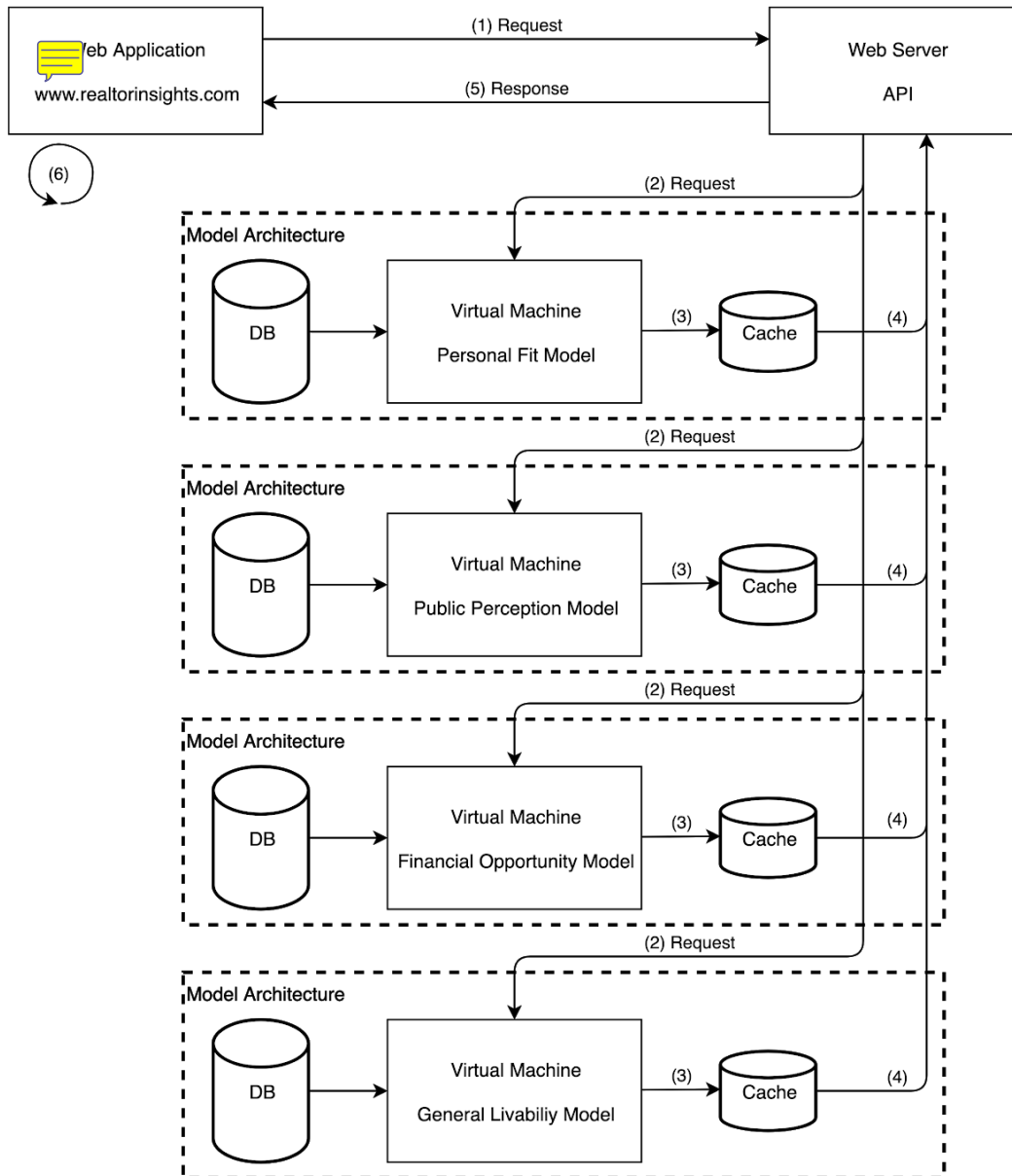
For the purposes of analyzing the feasibility of Assessor, the TELOS feasibility model is leveraged [4]. **TELOS** is a mnemonic for Technology, Economic, Legal, Operational, and Scheduling feasibility, and it helps determine the specific needs to implement a particular system. In Technological feasibility, 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 responsibilities) 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 [5][6]:

- 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 (for example, realtorinsights.com - domain to be confirmed upon creation of aws accounts) 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 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 [7].

### 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 (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) or Amazon DynamoDB (no-SQL). The cache will be hosted on Amazon S3.

## 2.1.2 Personal Fit Model

### 2.1.2.1 Introduction

The Personal Fit model implements a classification algorithm to recommend the user a neighbourhood given one's information. Since the residents currently living in a neighbourhood chose to live there and are continuing to reside in it, the Personal Fit model attempts to classify an individual to a neighbourhood which has its average resident most closely resembling them.

### 2.1.2.2 Data Sets

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 [8]. However, we could not easily find information regarding the relationship between the census tract and hood number or neighbourhood name [9]. 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 [10]. 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 Implementation

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 [11]. 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 [12]. 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 [13]; in other words, it does not require any training.

To implement the KNN algorithm, the following steps will need to be taken [14]:

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 [15]. 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.

#### 2.1.2.4 Evaluating Success



#### 2.1.2.5 Risks and Mitigation Strategies

### 2.1.3 Public Perception Model

#### 2.1.3.1 Introduction

The Public Perception model crowdsources information from social media and news websites to consolidate an image of how the public views a neighborhood. Often times, home buyers are predisposed to prefer certain neighborhoods over others due to things they hear online, or via word-of-mouth. This model seeks to address this intangible component of the purchase decision, pulling together both subjective and objective textual content. Various techniques in natural language processing (NLP) are applied on the data to extract quantifiable insights that aid in predicting what neighborhoods are thought of as more favorable by the public.

#### 2.1.3.2 Sources of Data

As the model concerns text, we consider sources where users share textual data. We find an abundance of subjective textual content on social media - hence, we plan to collect data from Twitter and Reddit. Both websites have standardized APIs for pulling data from its platform, detailed in [16] and [17], respectively. On Twitter, we will harvest data daily via keyword search, looking for tweets that concern a neighborhood. While the approach is subject to change, an initial approach may be to search for tweets containing the name of the neighborhood of interest, alongside the city name. On Reddit, we will retroactively collect data from relevant subreddits (such as the Toronto subreddit [18]). With subjective data represented, we turn our attention to objective data. While all text based data is inherently biased in some respects, fact reporting journalists attempt to maintain an objective viewpoint in their work. To collect news data, we look at reputable websites most likely to address events that occur in the city of interest. Our current plans are to look at CNN and the New York Times as a starting point. These websites also have standardized APIs for retrieving news content (detailed in [19] and



[20], respectively) - however, there is an application process in place to ensure that API usage complies with their rules. While we believe it is likely we will be granted access, we have alternate plans to crawl other news platforms in a way that complies with their terms of services (indicated in their robots.txt file).

### 2.1.3.3 Producing Quantifiable Metrics using 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. With regards to implementation, we will first replicate proven models in literature including the Stanford NLP model [21] and the VADER model [22]. 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 [23]). 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.

### 2.1.3.4 Extracting Key Topics from Text using 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 representation. With regards to implementation, we plan to leverage both the Latent Dirichlet Allocation (LDA) model [24], as well as the Bi-term Topic Modeling (BTM) model [25]. 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.

#### 2.1.3.4 Building a Data Pipeline

We recognize that the public perception model involves a lot of moving pieces - from harvesting data, to running the two disjoint analyses. Furthermore, we note that the data sources update on a second-level basis, requiring our application to be responsive to frequent updates in data. To address this, we plan to develop a concrete data pipeline, from its ingestion to its analysis. More specifically, VMs (AWS EC2 instances) will be always listening for updates from the APIs, and pull them into cache (AWS S3) once it receives them. The addition of new data will trigger an update for the two models, which will subsequently assign a sentiment score and topics to the new data. These will be pushed to our database, ready for the application to render the data. We hope to achieve turnaround time in the scale of a few minutes from data ingestion to availability of analysis.

### 2.1.4 Financial Opportunity Model

#### 2.1.4.1 Introduction

The Financial Opportunity 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 [26]. In the context of this model, the variable is median sale price of all homes in a district. 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 Data Set

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 [27]. The Financial Opportunity model is interested in the median sale price by neighborhood, but can be adapted to analyze other data sets.

#### 2.1.4.3 Approach

One approach to time series analysis is to use statistical methods to model the data. The median sale price of all homes in a district is an example of a univariate time series. A univariate time series is defined as a time series that consists of single (scalar) observations [28]. Common approaches to modeling univariate time series include triple exponential smoothing and Box-Jenkins approach [29]. 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 [30][31]. The ARIMA model differs from the ARMA model in that non-stationary time series are differenced one or

more times to achieve stationarity [32]. The Box-Jenkins approach assumes stationarity, so the model will be ARMA or ARIMA [32].

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 [33]. A seasonal time series is defined as a time series with periodic fluctuations [34]. Stationarity can be determined from a run sequence plot, and seasonality can be determined from an autocorrelation plot [35]. Once stationarity and seasonality are addressed, autocorrelation and partial autocorrelation plots assess the order of the autoregressive and moving average terms [35]. The estimation stage aims to fit the Box-Jenkins model by estimating the parameters. Maximum likelihood estimation is generally the preferred technique [36]. 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 [37]. 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 [38]. LSTM is a model organized into cells that passes an internal state variable from one cell to another [38]. The internal state variable is modified by operation gates: forget gate, input gate, and output gate [38]. LSTM models learn which past data is important in making future predictions [38]. 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.

#### 2.1.4.4 Implementation

To perform time series analysis on historical median sale prices by district, 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 and machine learning tools, which can implement the Box-Jenkins approach and LSTM model. However, 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.

To measure the correctness of the Financial Opportunity model, this model will 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. The performance of

this model is not relevant, as the time-consuming analysis is done beforehand, not when a user requests valuation forecasts.

## 2.1.5 General Livability Model

### 2.1.5.1 Introduction

The General Livability model is intended to compare any particular district to the ideal district - the “best place to live”.

Determining the ideal district is a difficult thing to define; 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 foreign assignments to appropriately price compensation incentives [39].

### 2.1.5.2 Approach

This model will leverage the indicator data collected by Mercer (stability, healthcare, education, etc.) for each city currently ranked on the Mercer ranking to learn what makes a city “ideal”. The underlying model will be a neural network, which is an unsupervised machine learning model. In this system, a model which learns to understand characteristics about the data it processes is trained, and when shown new inputs, classifies them using those same “self-taught” observations. Evaluating the model’s effectiveness may be achieved by processing the previous year’s city rankings, and determining the accuracy compared to actual rankings.

### 2.1.5.2 Data Set

Gathering the required data will be done by either contacting Mercer to determine the costs associated with sharing their collected data, or by mining the data (the key indicators for a given year) for each country on the Mercer list from the internet. Given that the datasets will be expensive, the mining approach will likely be taken.

Once trained and capable of accurately classifying any city given its indicators, the General Livability model will help determine a districts “livability”, which in turn will help the individual make a more informed purchase decision.

## 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 [40]; while the EC2 instance costs US\$0.006 to US\$7.00 / hour [41], depending on how powerful the compute server is required to be. For the databases, RDS will cost anywhere from US\$0.04 to US\$9.00 / hour (heavy storage requirements) [42], while DynamoDB charges per-write request, at US\$1.50 / million writes + US\$0.30 per GB / Month for storage [43].

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. 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.

For the Personal Fit model, data about the individual 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 2 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, and will be able to setup the required system architecture without too much difficulty.

### 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 2a: 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 1b: 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.

## 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.

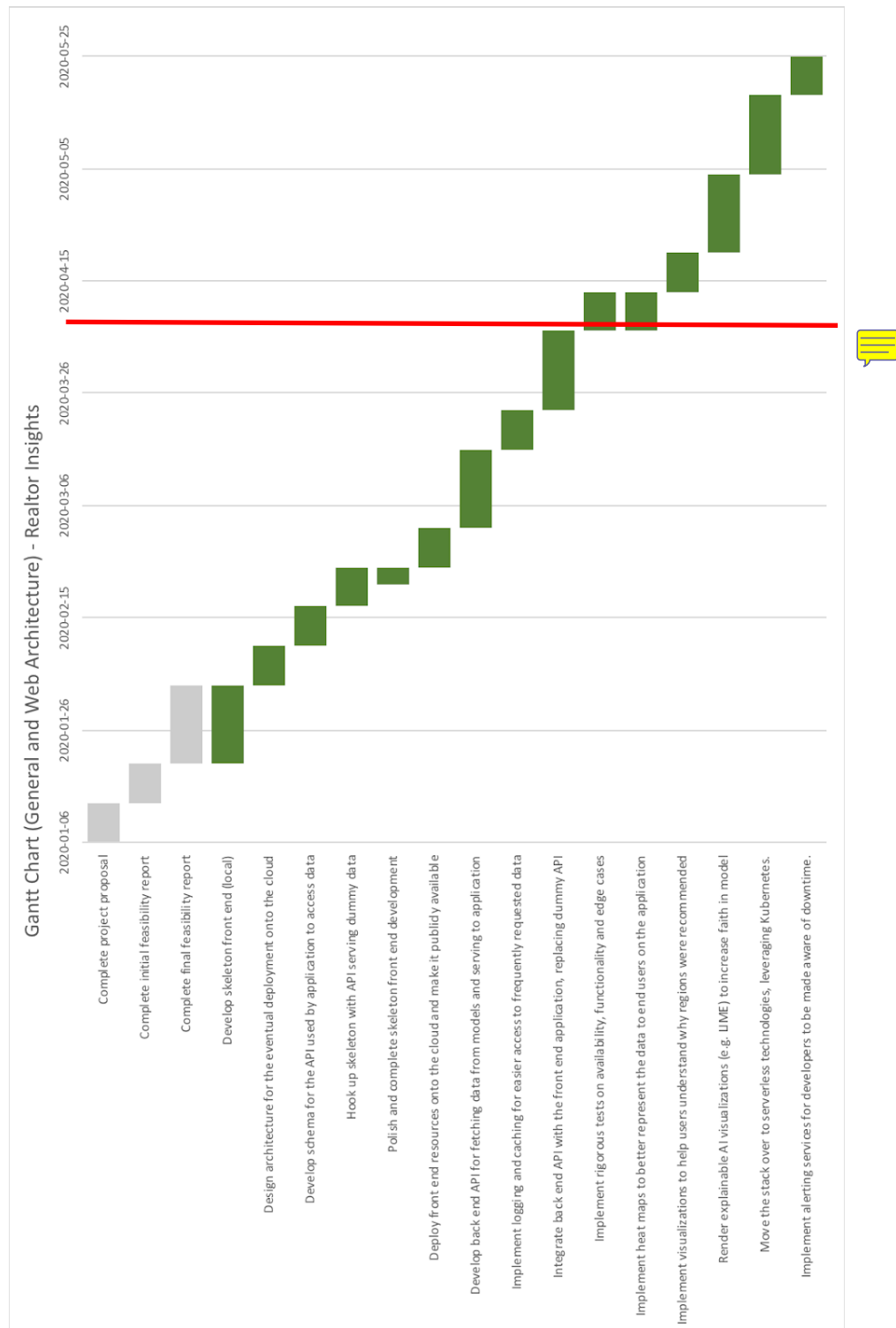


Figure 2a: Sprint breakdown concerning administrative in gray and system architecture in green

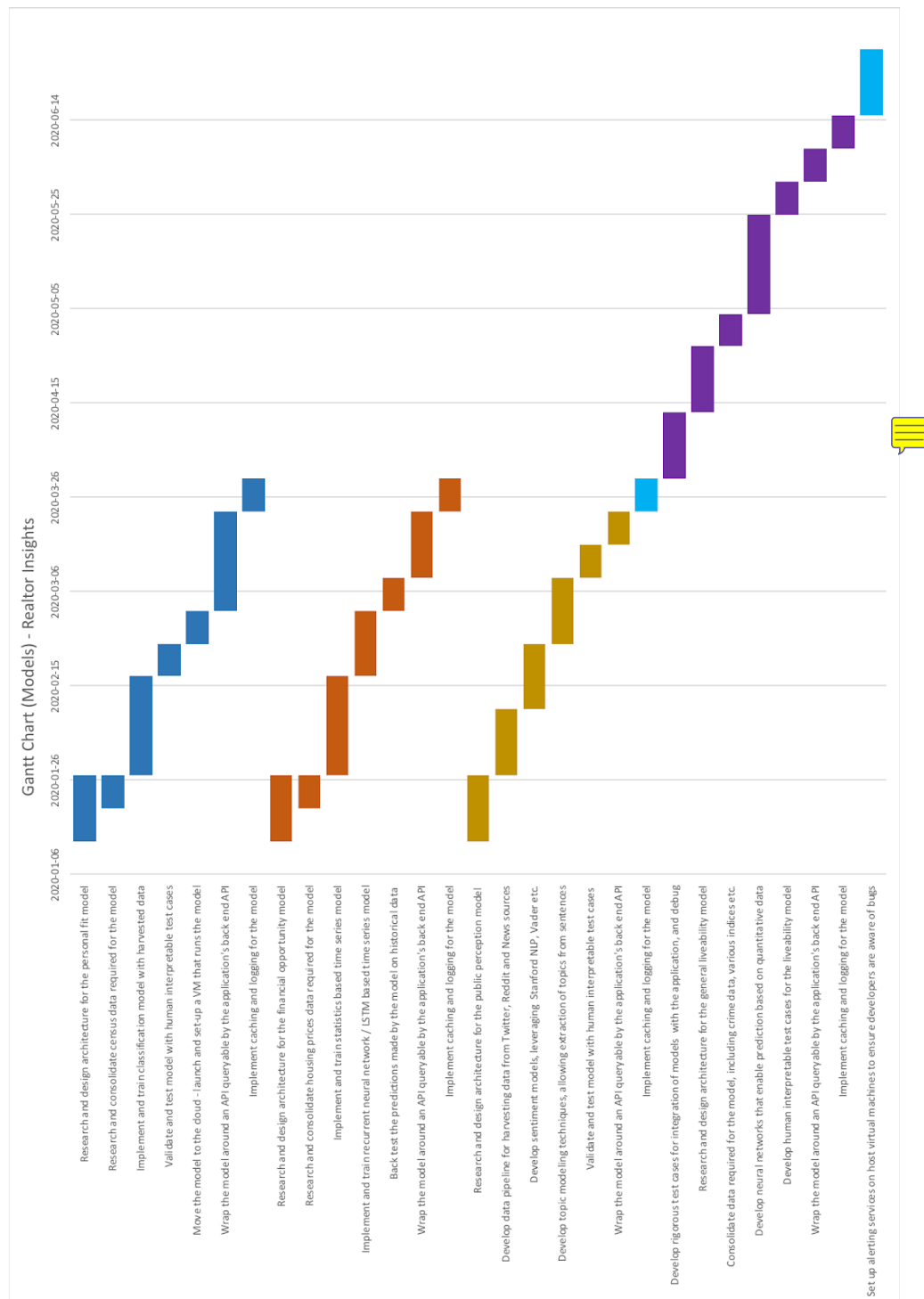


Figure 2b: Sprint timeline concerning personal fit model in blue, financial opportunity model in orange, public perception model in brown, general liveability model in purple, all models in light blue

### 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 (with the exception of heatmaps) are out of scope (Figure 2). 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] P. M. Heathcote and S. Langfield, A level computing. Ipswich: Payne-Gallway, 2004.
- [5] “S3,” Amazon, 2002. [Online]. Available: <https://aws.amazon.com/s3/>. [Accessed: 21-Jan-2020].
- [6] D. J. Daly and D. J. Daly, “Economics 2: EC2,” Amazon, 1987. [Online]. Available: <https://aws.amazon.com/ec2/>. [Accessed: 21-Jan-2020].
- [7] “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].
- [8] 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=IwAR3XBIB1X07fHAQMvTCYhdzsbajypqxrJlpW-m-zCq-q8eIW13G7qLStcX4>. [Accessed: 21-Jan-2020].
- [9] 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=IwAR00Ei1tFe5iSm4GSd8WUee\\_3uzGrAI9A9ZGR6wC5srL9WK2CY4RBoMth4](https://www.toronto.ca/city-government/data-research-maps/maps/census-tract-reference-maps/?fbclid=IwAR00Ei1tFe5iSm4GSd8WUee_3uzGrAI9A9ZGR6wC5srL9WK2CY4RBoMth4). [Accessed: 21-Jan-2020].
- [10] 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](https://data.cityofnewyork.us/City-Government/Demographics-and-profiles-at-the-Neighborhood-Tabu/hyuz-tij8?fbclid=IwAR24-fawwhoLS_meww5YIrQoEbuVE23vsqqWIB5171dKmHdBt3Hm0LF1734). [Accessed: 21-Jan-2020].
- [11] 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].

- [12] 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].
- [13] "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].
- [14] 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].
- [15] "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].
- [16] "Docs - Twitter Developers," Twitter. [Online]. Available: <https://developer.twitter.com/en/docs>. [Accessed: 21-Jan-2020].
- [17] reddit.com: api documentation. [Online]. Available: <https://www.reddit.com/dev/api/>. [Accessed: 21-Jan-2020].
- [18] "r/toronto," reddit. [Online]. Available: <https://www.reddit.com/r/toronto/>. [Accessed: 21-Jan-2020].
- [19] CNN. [Online]. Available: <http://developer.cnn.com/>. [Accessed: 21-Jan-2020].
- [20] The New York Times. [Online]. Available: <https://developer.nytimes.com/>. [Accessed: 21-Jan-2020].
- [21] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. [Online]. Available: <http://nlp.stanford.edu:8080/sentiment/rntnDemo.html>. [Accessed: 21-Jan-2020].
- [22] 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.
- [23] Datasets. [Online]. Available: <http://www.cs.cornell.edu/home/llee/data/>. [Accessed: 21-Jan-2020].
- [24] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," Journal of Machine Learning Research 3, vol. 3, Mar. 2003.
- [25] 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
- [26] 6.4.1. Definitions, Applications and Techniques. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm>. [Accessed: 21-Jan-2020].
- [27] "Housing Data," Zillow Research. [Online]. Available: <https://www.zillow.com/research/data/>. [Accessed: 21-Jan-2020].

- [28] 6.4.4. *Univariate Time Series Models*. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc44.htm>. [Accessed: 21-Jan-2020].
- [29] 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].
- [30] 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].
- [31] 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](http://rstudio-pubs-static.s3.amazonaws.com/339483_08ba8b928339464e916e54901156b348.html). [Accessed: 21-Jan-2020].
- [32] 6.4.4.5. *Box-Jenkins Models*. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc445.htm>. [Accessed: 21-Jan-2020].
- [33] 6.4.4.2. *Stationarity*. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm>. [Accessed: 21-Jan-2020].
- [34] 6.4.4.3. *Seasonality*. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc443.htm>. [Accessed: 21-Jan-2020].
- [35] 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].
- [36] 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].
- [37] 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].
- [38] 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].
- [39] "Solutions," Quality of Living Data and Hardship Premiums for International Assignments. [Online]. Available: <https://mobilityexchange.mercer.com/quality-of-living>. [Accessed: 21-Jan-2020].
- [40] "S3," Amazon, 2002. [Online]. Available: <https://aws.amazon.com/s3/pricing/>. [Accessed: 21-Jan-2020].
- [41] 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].
- [42] "RDS: understanding animal research in medicine," Amazon, 2007. [Online]. Available: <https://aws.amazon.com/rds/postgresql/pricing/>. [Accessed: 21-Jan-2020].
- [43] D. Rangel, "DynamoDB: everything you need to know about Amazon Web Service's NoSQL database," Amazon, 2015. [Online]. Available: <https://aws.amazon.com/dynamodb/pricing/on-demand/>. [Accessed: 21-Jan-2020].