Buying vs. Renting a Property in Toronto - **Blueprint**

3 November 2025

Submitted by **Group 24:**

Matteo Van Der Plaat, 20mvdp, 20287556

Brandon Tate, 20bndt, 20277137

Daniel Heron, 21djh17, 20333945

Joel Prosser, 21jrp16, 20342170

Faculty Supervisor:

Mariam Guizani

Table of Contents

Table of Figures

Executive Summary

This project was designed to test and compare the performance of machine learning algorithms in forecasting the Toronto housing market. The objective is to determine the most accurate machine learning model at forecasting the market, compare it to previous solutions such as linear NPV calculators, and use its predictions to make a comparison against a linear model representing the alternative of renting a property and investing the surplus money into a stock market index over a period of 25 years, or the median amortization period of a mortgage in Canada. The minimum viable product is a fully optimized machine learning algorithm that is able to most accurately predict the Toronto housing market based on publicly available demographic, economic, and housing market datasets compared to other models. Additionally, if the MVP demonstrates a strong ability to predict market prices, the model will be applied to narrower segments of the data broken up by neighbourhood and property type (e.g. condo vs. semi-detached vs. single family homes etc.) to provide more insightful results.

The design specifications of this project require accurate data going back at least 15 years (3 predictions periods) to ensure minimum viable functionality, which will be defined by the model's ability to predict based on patterns in the data. With this data in hand, Support Vector Machines, Neural Networks, and XGBoost models will be trained and optimized to determine which individual or ensemble of models is most accurately able to predict market trends. From there, the model will be compared against previous methods to determine if it has more accurate predictions. Lastly, the output of this model will be compared against the linear models representing renting and investing in the stock market. This optimized model will be tested by training it on past data and predicting the market landscape of today.

With home prices in Toronto increasing by as much as 1100% in the last 40 years, many young people today have been priced out of the market and fear that they may never own a home. This project aims to provide students, young adults, and other prospective homeowners with more information and clarity on the economics of home ownership in today's economy as well as its alternatives.

1. Introduction

In 1985, the median detached home price in Toronto was just over $100,000 with a median family income of $30,000 after tax [1][2]. Most families with a reliable income and some savings would be able to afford a home or qualify for a good mortgage within the first few years of their working career. In 2025, the average detached home price in Toronto is just below $1,200,000 marking a 1100% increase in the span of a single generation [3]. In that same time span, the average household income in Toronto has increased by nearly 200%, to $85,000 annually after tax [2]. This disparity between the growth of the housing market and the growth of real household wages serves as motivation for this project.

The young adults of today are looking at a vastly different real estate landscape than their parents did at the same age several decades ago. The notion that homeownership is the holy grail of investments is finally being put into question. The bar to entry is so high now that people are choosing (or being forced into) renting instead of owning. While this might seem like a great loss to the generation that profited greatly from the real estate golden age of the last 40 years, those people seem to underestimate the strength of the stock market during the same time period. On January 1, 1985, the S&P500 was valued at a peak of 180 points [4]. On January 1, 2025, the same index was valued at a peak of 5935 points [4], marking an incredible 3200% increase – notably higher than that of the Toronto housing market. While volatility does play a part in investment decisions, the strength of the market cannot be ignored in this discussion. This project aims to provide an alternative to young adults feeling hopeless when faced with paying an average of nearly $1,000,000 for a 700-800 sq ft condo in downtown Toronto [5]. By taking years of historical data on housing prices into account alongside relevant data related to demographics and economics, this project aims to predict the landscape of the housing market 25 years from now. With this information in hand, prospective home buyers will be able to make an educated decision to build their wealth and maximize their well-being.

The timeframe of 25 years has been selected because it is the median amortization length of a mortgage in Canada [6]. The median mortgage term length is 5 years, meaning that interest rates and payment schedules are fixed on 5-year terms. The prediction period, defined as the time interval between predictions used to build the final forecast, will be set to be the same as the median term length of 5 years – this also coincides with the Statistics Canada census frequency of every 5 years. This means that 5, 5-year predictions will be calculated sequentially to determine the final 25-year prediction which will be used to create a final recommendation on renting vs. buying.

This document includes the methods through which the project will achieve this goal, and how it will be managed. The methods will be broken down into key design specifications, validation and testing of the model, and potential error points and their mitigation strategies. The sections describing the project management entail how project work will be divided among members, the schedule of project execution, the required budget, as well as progress to date. Finally, some important additional considerations will be covered such as how any health and safety risks will be mitigated, the environmental impacts the project may have, and any legal considerations that this project may have.

1.1 Design Problem

This project aims to solve an old problem in a new way, this problem being the question of whether buying or renting a property is more profitable long term. While this has been solved before with linear predictive methods, this project aims to tackle the problem with modern machine learning methods. While linear models are easy to implement, they provide simplified predictions that are not representative of real-world conditions. Machine learning algorithms are often able to provide more accurate models of problems of this nature due to their strengths in pattern recognition and generalization of large data sets, but with that comes increased technical challenge.

1.1.1 Data Constraints

While there are many data sets providing thousands of features and data points to train a machine learning model, the housing market is a notably difficult entity to predict due to its reactiveness to so many economic, demographic, and regulatory factors. This project aims to best predict the housing market with all available data; however, the scope of this project is quite large and it is unlikely that any available datasets are able to accurately and fully encapsulate the reasoning behind all movements in the housing market.

1.1.2 Model Optimization

Optimizing machine learning models is often a long and arduous process, this project will be no different. To optimize models, they must be trained and validated many times. This presents a technical challenge in selecting the correct hyperparameters to provide the most accurate output. Selecting and tuning hyperparameters requires a strong understanding of individual model strengths and intuition in interpreting results of each training and validation run.

1.2 System Specification

The system specifications are split into three different categories which are functional, interface, and performance. The functional specifications ensure that the project has a guided step-by-step system to follow and make sure it is completed properly. The interface specifications ensure that the input, output, and run parameters can all be modified and produce correct/accurate results. The performance specifications are for accuracy, runtime, robustness, and the pipeline success rate. Each of the three categories has its own table below to show each requirement, metric of success, and its acceptance criteria. The minimum viable product and the enhanced version of the project are listed in the performance specifications and requirements section (1.2.3).

1.2.1 Functional Specifications and Requirements

Requirement	**Metric / Unit**	**Target / Acceptance Criteria**
Input historical housing & economic data (CMHC, StatsCan, BoC, City of Toronto)	Successful data import rate	100 % of selected tables loaded with no errors
Preprocess & integrate datasets (cleaning, combinations, normalization)	Pipeline completion rate	100 % pass on curated dataset; ≤ 5 % missing features after cleaning

Requirement	Metric / Unit	Target / Acceptance Criteria
Train three models (SVM, Neural Network, XGBoost)	Successful training runs	All three models converge without runtime errors
Generate 25 year forecast by neighbourhood and property type in 5 year intervals	Forecast horizon (years)	≥ 5 years baseline forecast (5 intervals)
Export predictions for UI integration	File formats	Excel or CSV export validated for neighbourhood, price, and year
Scenario forecasting	Scenario types	Baseline, optimistic, conservative

1.2.2 Interface Specifications and Requirements

Requirement	Metric / Unit	Target / Acceptance Criteria
Input feature file contract	Column validation	Columns, data types, and units match expected values
Output prediction schema	Structure to the outputs	neighbourhood, property type, year, predicted price
Configurable run parameters (optional)	Parameter format	JSON configuration loadable without code changes

1.2.3 Performance Specifications and Requirements

Requirement	Metric / Unit	Target / Acceptance Criteria (Minimum Viable Product vs Enhanced System)
Forecast accuracy (primary performance requirement)	Root Mean Squared Error (RMSE),	MVP: RMSE < 10-20% of the target price; Enhanced: RMSE < 5-10% of the target price
Comparative baseline improvement	Percent reduction in RMSE vs. ARIMA / Linear Regression	MVP: 3% lower RMSE than Linear Regression models

Requirement	Metric / Unit	Target / Acceptance Criteria (Minimum Viable Product vs Enhanced System)
		Enhanced: 10% lower RMSE than Linear Regression models
Runtime efficiency (training)	GPU utilization	MVP: ≥ 80% GPU usage Enhanced: ≥ 90% GPU usage
Robustness	Prediction stability under ±5% input changes	MVP: Change in output ≤ 10% under controlled input variation Enhanced: Change in output ≤ 5% under controlled input variation
Scalability	Runtime growth with increased dataset size	MVP: Doubling data causes ≤ 100% increase in training runtime Enhanced: Doubling data causes ≤ 50% increase in training runtime

2. Methodology

2.1 Approach

This section entails the design approach of the project and the sequence of steps to be followed, which are outlined below in 2.1.1 – 2.1.5. The overarching design approach follows an agile development framework where sprints of development will be completed and then reviewed alongside the project supervisor and TA to ensure all analysis is valid and correct. These will be managed using project management programs such as Trello and Jira to track progress and ensure accountability among project members. The sprints will be broken into the five sections outlined below, with feedback incorporated all throughout, to achieve a fully functional and optimized end product. How each sprint will be broken up and how modular design will be incorporated is outlined in the corresponding subsections below. Iterative prototyping will be managed through the use of GitHub. GitHub allows for each iterative model to be stored alongside detailed documentation to ensure organization of previous versions. The documentation attached to each model iteration will detail its specifications as well as its performance.

The step-by-step development approach is detailed below. Figure 1 visualizes the execution of the steps described below.

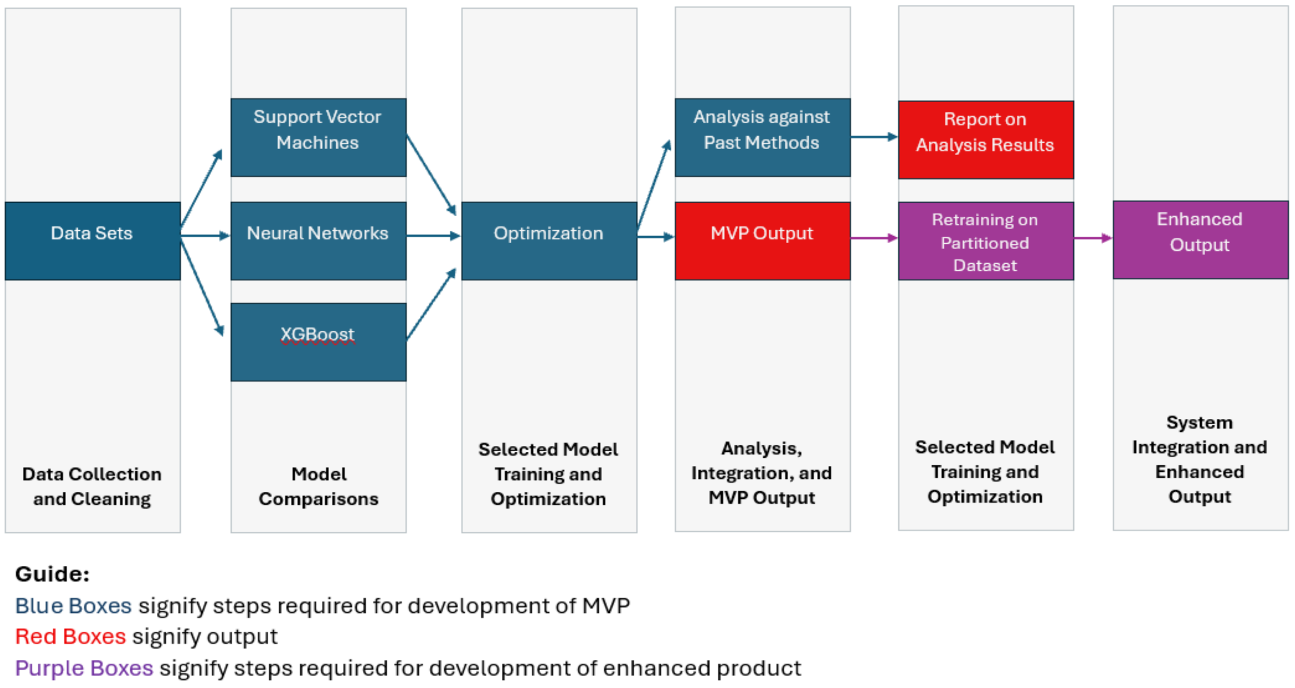
Data Sets
Data Collection and Cleaning
Support Vector Machines
Neural Networks
XGBoost
Model Comparisons
Optimization
Selected Model Training and Optimization
Analysis against Past Methods
MVP Output
Analysis, Integration, and MVP Output
Report on Analysis Results
Retraining on Partitioned Dataset
Selected Model Training and Optimization
Enhanced Output
System Integration and Enhanced Output
Guide:
Blue Boxes signify steps required for development of MVP
Red Boxes signify output
Purple Boxes signify steps required for development of enhanced product

Figure 1: Project Approach Diagram

2.1.1 Data Collection and Cleaning

The first preliminary step to training any machine learning models is collecting large, detailed datasets. For this project, many datasets have already been uncovered in preliminary research, however, all members will engage in a short sprint to get as much more relevant and reliable data as possible before beginning the model comparisons phase where all models will be trained on this data. Datasets will also be cleaned for irrelevant or empty data and put into an optimal format for consumption by machine learning models.

2.1.2 Model Comparisons

In this phase each model outlined above (Support Vector Machines, Neural Networks, XGBoost) will be trained on the data from the data collection phase. To ensure modular design, each model will be individually optimized and trained to the best possible performance during the sprint. By the end of the sprint, the model with the best overall performance will be selected as the model to be fully optimized. Performance will be evaluated by judging each model's accuracy at determining today's housing market prices based on historical data.

2.1.3 Selected Model Training and Optimization

Once the best performing model has been selected from the prior phase, further intensive optimization will be the focus of the next sprint. All testing will be saved and documented in GitHub as all test scenarios help the model get closer to peak performance, even if individual tests result in poorer performance. To ensure efforts are not duplicated, the first phase of this sprint will be dedicated to researching the best ways to optimize the selected model. From there, group members will optimize, train, test, and upload the model alongside detailed documented to GitHub to follow the principles of modular design and allow for comparison and a potential combination of optimizations.

2.14 Analysis

Following the optimization of the model, its performance will be compared against previous methods to demonstrate the effectiveness of newer machine learning methods at modelling the housing market. If the model fails to outperform past methods, it will demonstrate that machine learning models are not yet able to predict large, highly fluctuating entities such as the housing market, however, it will put forward the best machine learning model to be used for future analysis of potentially less complex market entities than the housing Toronto market.

2.1.5 System Integration and Output

With an optimized machine learning model in hand, this sprint will be dedicated towards integrating the model into a system that answers the original question of whether renting or buying a property in Toronto is more profitable. For comparison against model predictions, a linear model of renting and investing in the stock market will be developed from Toronto rental market and stock market index averages. Finally, the output will be developed to display the results of comparison between model predictions and the predictions of the linear model representing renting a property and investing in the stock market. This sprint may have several different outcomes depending on the model performance determined in previous sprints. Nonetheless, results will be displayed, and a final conclusion will be drawn on a.) which machine learning algorithm performs best at predicting highly dimensional, complex entities such as the Toronto housing market, b.) whether machine learning models are fit to predict such complex and unpredictable entities such as the Toronto housing market, and c.) whether buying or renting a property is more profitable in a 25 year time span in Toronto. This final comparison between renting and buying will be made using the following equations:

$$\begin{aligned}(1)\ Buy = {} & predicted\ median\ property\ price - current\ median\ property\ price \\ & - (associated\ annual\ costs\ of\ homeownership * 25\ years)\end{aligned}$$

$$\begin{aligned}(2)\ Rent = {} & (income\ freed\ up\ by\ renting\ instead\ of\ buying * 25\ years \\ & * compounding\ average\ nominal\ yearly\ RoR\ of\ S\&P500) \\ & - (current\ median\ annual\ rental\ price \\ & * maximum\ annual\ rent\ increase * 25\ years) \\ & - (yearly\ associated\ costs\ of\ renting * 25\ years)\end{aligned}$$

‘*Income freed up by renting instead of buying’* will be determined by subtracting the average cost of rent in Toronto from the median monthly mortgage cost. The Future Value (FV) formula to consider compounding interest is available in the appendix. This will be used to more accurately calculate returns generated by the stock market index and rental prices, as seen in the ‘*rent’* equation. Annual associated costs for the ‘*buy*’ option may include property tax, upkeep, utilities, etc. Associated costs for the ‘*rent’* option may include utilities, renter’s insurance, etc. Further in-depth research will be performed in this phase to ensure an accurate comparison between buying and renting. All values will be calculated in nominal terms with inflation factored in where applicable.

If the minimum viable product is achieved, meaning the model is demonstrated to be able to forecast the housing market with higher accuracy than past solutions by recognizing patterns in the data, the output will be expanded to display information by property type and neighborhood in Toronto. This information is all available in previously discovered data sets, however the sectioning of the data into many neighborhoods and several property types will present a technical challenge in training the model due to the division of a few large data sets to many smaller data sets.

2.2 Design Tools, Hardware, Instrumentation

2.2.1 Previous Non-Machine Learning Solutions

This problem has been solved in the past without using machine learning. It is important to understand these previous solutions to see which methods have shown success and how machine learning can be used to improve these previous solutions. The main solutions to this problem have been net present value math, cash flow analysis and ARIMA models.

The New York Times has a net present value ("NPV") calculator in which you plug in known values about a property for sale, as well as some values for a property on the rental market, and it can calculate what the most profitable option would be for individual cases. Some of these criteria include home price, rent, mortgage rate, investment return on savings (if you rent, what do you do with the money saved?). This method of solving the problem is useful for individual cases where users are trying to decide between two comparable properties in the present moment, but does not consider future economic trends, housing market trends, demographic trends, etc. The goal of this project is to take that guess work out of the calculator and attempt to predict market trends to provide a more accurate, holistic prediction – not just an NPV calculation.

AutoRegressive Integrated Moving Average ("ARIMA") models are another solution to this problem. The Canadian mortgage and housing corporation use models like these to provide answers however they are better at solving much shorter time periods (1-3 years in most cases). These models are used because they work with smaller datasets, they handle seasonality quite well (Canadian housing is very seasonal) and they can give you accurate results without having to use more advanced machine learning. There are also limitations to these models. ARIMA models assume that there are linear relationships however the housing market is influenced by many factors that are complex and often non-linear. Things like interest rates and immigration affect housing prices, however they are non-linear relationships.

Our goal is to create a model that will outperform current methods by implementing machine learning algorithms to predict market trends. The next section will include the methods that can be used to solve this problem.

2.2.2 New Machine Learning Solutions

There are multiple machine learning algorithms that can be used to solve this problem. Some of the more applicable ones include Support Vector Machine ("SVM") models, Neural Networks and Extreme Gradient Boosting ("XGBoost"). These algorithms handle high dimensionality, and they are also highly researched. Since they have been used for other similar projects like predictions in mineral deposits for mining, the results from those projects can be used to help create a solution to this financial analysis.

SVM models are useful when trying to understand a smaller dataset. The datasets from the Toronto market are considered smaller datasets in terms of machine learning (thousands of data points instead of millions). SVM models are also efficient at understanding highly dimensional data. Extra dimensions or features like population and immigration will be implemented into the datasets by combining multiple datasets and quantifying certain categories. Some examples include month of year / season and unexpected events like COVID or stock market crashes and fluctuations. SVM models are also easier to tune than many other machine learning algorithms like neural networks. The only problem with SVM models for this project is the fact that when making predictions the data point will be given a class prediction. Therefore, instead of an exact prediction of price it will have to be a range between \$50 000 – \$100 000. Another fix for this problem is to use Support Vector Regression ("SVR") instead of a more standard Support Vector Classification ("SVC"). KNIME will allow different outputs for a SVM including both regression and classification. For this project both models will be tested with comparing results. A screenshot of an example architecture diagram is shown below.

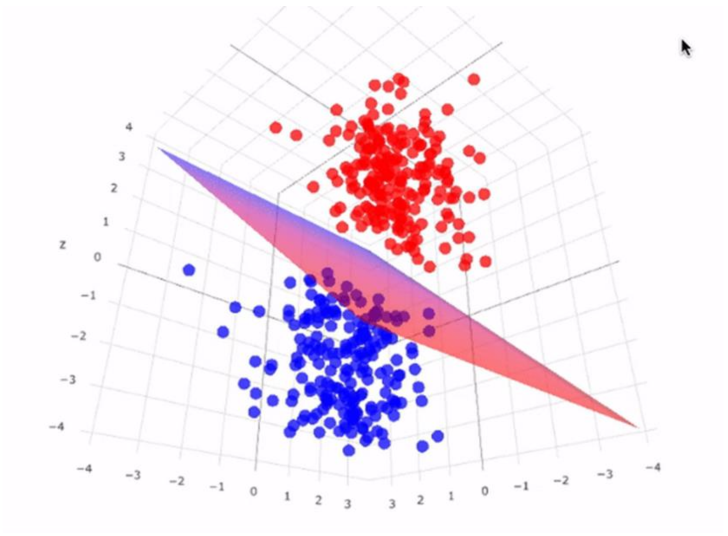

Figure 2 - SVM Architecture Example

This architecture diagram shows how SVM models classify different data points. The hyperplane changes based on features and the outputs of the training data. The best hyperplane corresponds to the midline of the thickest surface that can fit between datapoints of the classes. For this project there will be more classes to fit all of the price ranges needed for the predictions.

Neural Networks work with larger datasets, however are still accurate with smaller datasets. Neural Networks also gain advantages with high dimensionality. By adding more information about each individual piece of data you have the neural network can begin to determine which features are useful to the model. This may cause certain features to have greater impact than was initially thought by economists. Weights inside of the neural network will change when the model is training which is what causes certain features to have varying levels of impact on the output. Each feature is assigned an initial weight and by accessing each of the weights you can then determine the value of each feature. Although these models are more complicated, they will generally return better results if they can be tuned properly. Unlike SVM models Neural Networks will be able to make an exact prediction of the price since the results do not have to be tied to a class (although they can be). A screenshot of an example architecture diagram is shown below.

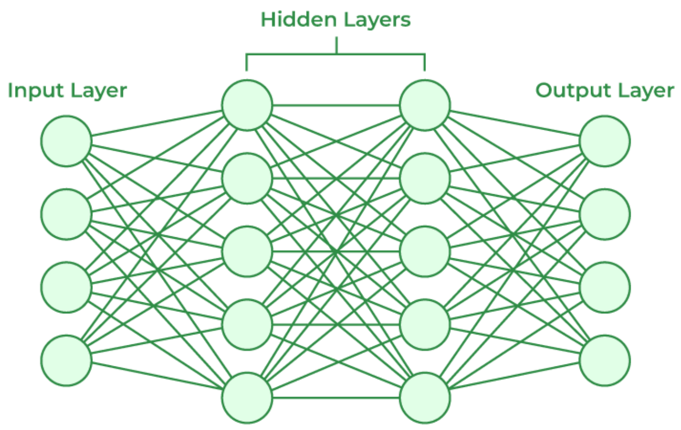

Figure 3 - Neural Network Architecture Example

There are three main aspects of the Neural Network. Those are the input layer, hidden layers and the output layer. The input layer is where the data enters the network. Each input node corresponds to one feature of the data. The hidden layers perform computations and try to extract patterns from the input layer. This is where weights are changed as explained before. Finally, the output layer produces the final prediction of the network. This problem will involve regression output and a multi-class classification. The regression output will give an exact house price estimate, and the multi-class classification will give a range like the SVM model.

XGBoost is a machine learning algorithm that builds multiple decision trees to make predictions [7]. Gradient boosting refers to starting with a single tree and continuing to add new trees to the model. Every new tree added attempts to fix the problems of the previous tree. The combination of these trees gives you your final prediction. This algorithm is good at understanding non-linear relationships. Many of the features used in this solution will be non-linear and therefore this is a potential model to solve the question.

2.2.3 Software, Development Tools, and Data Resources

Category	**Tools**	**Purpose**
Operating Systems&	Windows 11, Python	Windows for running KNIME workflows, Python scripts, and Colab notebooks. Python is the

Category	Tools	Purpose
Programming Environments		chosen programming language for its simplicity and ready to use machine learning libraries.
Frameworks& Libraries	TensorFlow, Keras, XGBoost, Pandas, NumPy, Matplotlib	Core libraries for building, training, tuning, and visualizing the machine-learning models.
Development Tools & IDEs	KNIME Analytics Platform, Google Colab, VS Code	KNIME handles data-flow design and feature engineering; VS Code supports code versioning and debugging to then be used in Colab for training; Colab provides GPU-accelerated training.
Version Control & Collaboration	GitHub Repositories, Teams	GitHub keeps reproducible code versions; Teams for documentation and communication.
Simulation& Testing Tools	Jupyter Notebooks through Colab, Excel Spreadsheets	Jupyter Notebooks will be used after the VS Code versions have been completed; Excel spreadsheets will be the output of the KNIME workflow.
Data Resources	Statistics Canada Census Data, CMHC Housing Reports, Bank of Canada Financial Indicators, City of Toronto Open Data Portal	High-credibility, publicly available sources that provide economic, demographic, and financial features necessary for forecasting. These datasets will be used for the training of the models.

KNIME will be the initial workflow because of its node connections and simple use of machine learning algorithms. The datasets will be combined and cleaned in an excel file that can then be read into KNIME through an excel reader node. Normalizer nodes are the next step to ensure accurate results when using Support Vector Machine, Multi-Layer Perceptron, and decision tree nodes. Visualization nodes will also be used to visualize the predictions and see how accurate the predictions are.

TensorFlow and Keras will be used for the Google Colab section of the project. Keras is an API built on TensorFlow which will allow a simplified process to build the neural networks without having to sacrifice the flexibility of parameters. These models will be more complex than the KNIME models and should allow for higher accuracy because of the parameter flexibility.

2.2.4 Hardware

Category	Equipment	Purpose
Local Hardware	Intel i9 CPUs, 8GB NVIDIA RTX A2000	Used for dataset preprocessing, feature selection, and smaller models. May also run larger models to see differences in length of training. KNIME models will run through this local hardware
Cloud Resources	Google Colab Pro GPUs (T4 / A100)	Used for neural-network training and hyper-parameter optimization requiring longer computing times.
Testing / Benchmark Instrumentation	Python Profilers and Timing Utilities	Evaluate execution time and training efficiency for scalability and accuracy analysis.

The python versions of the models will be run through local hardware as well as Google Colab. The local hardware may not be fast enough to train the complex models in a reasonable timeframe which is why the Google Colab subscription will be required. The complex models will be trained through Google Colab's cloud gpu's in reasonable time frames to then be compared against multiple models and see which ones are the most accurate.

2.3 Validation and Testing

2.3.1 Data Validation

Data validation will be done in a few different ways. The first is through KNIME's column filter, and data type converter nodes. The initial imported dataset will be from an excel file which needs to be modified before it can be used for training. The column filter node is used to select and deselect the columns that are to be used for training models. The datasets being used will have information that will not help improve accuracy for this project and will need column filter nodes to deselect that information. The data type converter is used to switch the data types of certain columns of information. When the data is read from excel the data types may come in incorrectly. For example, the house prices may come in as strings instead of integers and if that change to integer is not made before the model is trained then the information can't be used therefore making the data useless.

The next use of data validation is for outliers. By using z-score certain anomalies can be normalized to a closer standard deviation. Feature normalization is also another way to use data validation, and it uses the same normalization as the outliers (z-score). The normalizer node in KNIME allows a certain data type to be normalized. The normalizer node ensures that all the values have similar ranges without distorting the differences between the ranges. This is needed to ensure the higher values do not dominate the model's training and learning process. Although house prices are unlikely to have outliers it is still important to validate the data through different techniques to ensure the highest accuracy when training models.

2.3.2 Model Validation

Model validation will be done through cross-validation, regularization and benchmark comparison. Cross-validation can further enhance the learning process of the model by repeatedly splitting the dataset into training and testing sets. Cross-validation helps to get a more reliable estimate of how your model will generalize to new data. By doing this you can avoid overfitting.

Regularization is similar to normalization from the data validation section which also reduces overfitting. The benchmark comparison is done by comparing the results of the

ARIMA and linear regression models that already exist with the results of the new machine learning models. This not only tells you if the models are accurate but it also tells you if the machine learning models are more accurate than the previous models being used.

A validation summary table is below to show every validation method being used and the target values for this project.

Requirement / Spec	**Validation Method**	**Metric**	**Target / Acceptance Criteria**
Data Completeness	KNIME Missing Value Check	Proportion of missing data entries (% Missing)	No more than 5% of data missing after cleaning (≤ 5.0 %)
Feature Scaling	Z-Score or Min–Max Normalization	Distribution of feature means and standard deviations (μ, σ)	Average values centered around zero (μ ≈ 0) with variation within 10% tolerance (σ ≈ 1 ± 10 %)
Support Vector Machine	10-Fold Cross-Validation	Prediction error (RMSE) and goodness of fit (R^2)	Prediction error below b (RMSE < 0.30) and fit score above 0.80 ($R^2 > 0.80$)
Neural Network	80/20 Train–Test Split	Average error (MAE, MSE) and training–testing loss difference (ΔLoss)	Average error below 0.25 (MAE < 0.25) and loss gap under 0.10 (ΔLoss < 0.10)
XGBoost	Grid Search and Cross-Validation	Prediction error (RMSE) and model efficiency	Prediction error below 0.25 (RMSE < 0.25) and efficiency improvement of at least 10% compared to baseline

2.3.3 Integration and System Testing

After validation the KNIME and Colab models will undergo testing to confirm stability and performance. This includes performance testing and robustness testing. Performance testing measures the GPU utilization and runtime performance from both the local hardware and the cloud hardware. This is helpful if you get similar results between models, however one model uses a certain percentage more to complete its training and testing. Robustness testing is to be used to ensure the models are generalizing house prices properly. This testing gives stronger conclusions that the model performance is consistent even with slightly inaccurate data.

2.3.4 Post Training Error Analysis

The post training error analysis is done by looking at the weights in the neural network models. Each of the features are assigned weights when the neural network is learning. Features of the neural network include house prices, number of bedrooms, number of bathrooms, location, etc. Once the model has finished learning you can determine the most important features that help predict future house prices by seeing how much each of the weights affect the output. These weight values can be found in KNIME and will then be used to help build a more accurate python version.

This testing summary table shows target values for the 3 post training tests that will be tracked for this project.

Test Type	**Method / Tool**	**Metric**	**Target / Acceptance Criteria**
End-to-End Pipeline Test	Full KNIME and Colab workflow execution	Workflow completion rate / data transfer integrity	Successful execution in 100% of runs with no data loss or file corruption
Performance and Scalability Test	TensorFlow and Colab runtime analysis	GPU utilization, training latency, and runtime efficiency	Maintain at least 85% GPU utilization and consistent runtime performance across runs (±10% variance)
Robustness Test	Input changes and noise injection	Prediction variance and stability of outputs	Changes in input features of ±5% should cause less than 5% deviation in predicted outcomes

2.4 Problems and Mitigation

Throughout the project there are expected to be problems and delays which throw off the timeline, the following are the expected problems and mitigation strategies.

Google Collab server outage due to maintenance or unexpected. In such cases, Collab will be unable to access to train models.

Regular checking of the Collab website will be done so training time can be scheduled, not during maintenance. All code and trained models will be saved both locally and in a collab notebook, so in the case of an unexpected outage, training can be done on local GPUs.

Not all data is stored in the same way, so there may be issues when trying to format. In this case, the models may run into errors while training testing or validation.

To avoid this, a script will be made to scan the formatted dataset and ensure data points are correct. If a data point is incorrect, it will be printed, and a member of the team will attempt to fix it. If the data point cannot be fixed, it will be deleted. As another layer of protection, all scripts using the datapoints will implement a check and skip over any incorrect data points.

During the process of training our various models and neural networks, we are planning and expecting the training of these models to take a fixed reasonable time to train accurately and effectively. However, in some cases, training may take longer than previously foreseen.

To mitigate this potential risk, we have scheduled our training time to be completed well before the time needed for our final product to be finished as well as the showcase. The training process is currently timed to be finished around the end of January. This allows nearly 1.5-2 Months of runoff time if (but hopefully not) needed to retrain or continue training and fine tune the models for optimal performance and success. With total training time predicted to last a few weeks, an extra 6-8 weeks of run off time will be more than enough to work as a backup.

Upon the completion of the first iteration of model training, the expectation is for the network to be properly trained and to work precisely as expected and predicted in the blueprint functionalities. However, it is acknowledged and understood that when working with these technologies, this is not always the case.

This solution to this potential risk is the same as the previous case of training time. The initial training is to be fully completed no later than the end of January 2026. This will allow for nearly 6-8 weeks of run-off time before a final product is due to fix any potential performance errors that were unforeseen.

Once the initial training as well as other potential iterations are finished, the trained networks/weights will need to be stored somewhere other than just a co-lab file. This is as previous experience with co-lab has led to the loss of or improper saving of training files after hours of training are completed.

The solution for this problem is to be consistently monitoring the training when near completion, so the trained file can be immediately saved to another source. Multiple copies of the training file should be saved as well in case of accidental misplacement or deletion. This should include a physical location such as a laptop hard drive or USB drive as well as a shared and backed up team folder for easy access by the entire group.

Through the duration of the school year, there is a possibility of absences of various group members due to various reasons. This can cause group members to not be able to attend meetings, work on deliverables, or contribute to their assigned portion of the project temporarily.

To mitigate this risk many of the group meetings can be held online through Microsoft teams so that the absent member can still attend the meetings if their situation allows even though the member cannot attend in person. If a group member is unable to commit to their completion of a deliverable, other members of the group should be able to split the workload between themselves in order to finish the deliverables accurately and on time. Lastly, if a group member is temporarily unable to work on their assigned part of the overall project, the rest of the group members will have enough knowledge on the overall project to work on the group members' part until their return.

Problem	Mitigation	Recovery Strategy
Co-Lab Server Outage	Monitoring of Co-Lab, saving files locally and within the Co-lab notebook.	Training on local GPUs whist available.
Inconsistent data formatting	Create a script to find errors and inconsistencies in data.	Fix manually if possible or delete it.
Training time takes longer than expected.	Allow nearly 6-8 weeks of runoff time for training.	Use runoff time only if needed.
Performance of networks not reaching expectations.	Same as above	Same as above
Loss or improper storage of trained weights/networks.	Saving multiple copies on physical and saved drives.	Accessing the backup files either physically or on teams.
Group Absences	Move meetings online/split the workload of absent members.	Work on the absent members part until the group member returns.

3. Milestones/Division of labor

No.	Milestone	Due date	Responsible member(s)
1	Dataset collection	Week 3	All members
2	Data Formatting	Week 6	Matteo
3	KNIME implementation and testing	Week 8	Brandon
4	Collab implementation of top model from KNIME	Week 12	Matteo & Brandon
5	Collab implementation of second-best model from KNIME	Week 12	Joel

6	Collab implementation of third-best model from KNIME	Week 12	Daniel
7	Collab training, testing, validation & finetuning of the models	Week 16	All members, 2 solo 1 pair
8	Comparing and benchmarking models	Week 19	Matteo & Brandon
9	Visualization	Week 19	Daniel & Joel

4. Budget

4.1 Infrastructure Budget

Google Collab Pro monthly subscription for training and validating models estimated 2 months of subscription needed at $13.99 per month which will give 100 compute units per month and access to more memory on the server. To ensure enough compute units are available to complete the needed tasks, an extra 600 units will be needed, at the cost of $13.99 per 100 units. Total budget needed is $126.47 after tax.

5. Risk Mitigation, Environmental Impact, and Legal Considerations

The physical health and safety risks surrounding the project are minimal to near nonexistent. This is due to the project being completely software, and such software is not used on any physical equipment such as cars or medical equipment. Therefore, there is no error that can be produced that will cause physical injury or risk to a user using the program for their own use. However, there are always risks with using any electronics while coding software such as battery overheating and explosions or electrical shock from chargers. However, these risks are not unique to this project or software as these are general risks that can happen with any electronics. The environmental risks of the project are also somewhat minimal. Again, due to the project being strictly software, there will be no physical waste produced as well as no carbon or chemical emissions/waste. However, the one aspect taken into consideration is the training of the networks on an external server. The group is aware of the extreme amount of electrical energy and water sources needed to run remote GPU servers such as the ones Googles Co-Lab uses. Due to this, the group will be conscious of the amount of training being done and to make the effort not to train for longer than absolutely necessary for the project. Lastly, there are also not a great deal of legal ramifications within the project at hand. Again, due to the project not being anything physical or software controlling anything physical, the physical risks are non-existent. That being said, the group is making sure to state in the software and wherever else applicable that the software at hand is not to be used for legitimate financial advice. Any major decisions such as buying a multi-million-dollar property should be discussed with trained and professional financial advisors. Although the software at hand could be used as a tool to help gauge what certain investment outcomes *may* look like, the user should be well aware, no piece of technology, no matter how well trained, can accurately predict the future. The user should use the software with this in mind and at their own risk.

6. Progress to Date

As of October 24th, 2025, most of the progress completed on the project has been on planning, scheduling, and discussing how the project will be completed and how the idea of the project will become reality. The vast majority of that planning process and the outcomes of that process have been stated in various places throughout this blueprint. The group has a set schedule and process of how this project will be completed and the technologies and software that are going to be used. This process is scheduled to begin immediately after the submission of this blueprint deliverable. This first process will include gathering all necessary data from the datasets and scanning through potential errors with a script to fix or delete manually. The sources from which this data will be gathered have already been researched, allowing for a smooth and efficient transition to the next phase of the project.

7. Conclusion

For many Torontonians facing record-high property prices and stagnating wage growth, this project aims to provide information on a potential alternative to buying a home, which has become the status quo throughout the golden age of the housing market over the last 40 years [1]. The goal of this project is to provide a conclusive prediction, based on past and present states of the housing market in Toronto, what will be more profitable for the average prospective homeowner in 25 years – buying a property, or renting a property and investing extra funds into the stock market. Paired with some additional financial analysis such as cutoff points and percent certainty, the aim is to make a tool to better guide anyone considering entering the Toronto housing market.

So far, the required datasets have been found from trustworthy, publicly available sources such as CMHC, Bank of Canada, and StatsCan. These datasets each include thousands of data points with information on the Toronto housing market dating back as early as the 1950s. Additionally, the algorithms to be tested throughout the execution of the project have been selected based on preliminary research detailed above. Algorithms were selected for their ability to handle large, highly dimensional datasets, as will be required for this project. With the discovery of the required datasets, selection of the most promising algorithms, and the completion of the proposal document and TA meeting, this project is running successfully thus far and adhering to the schedule outlined in the project proposal.

References

[1] “Now and then: Do Canadian homes really cost that much more than 30 years ago?,” *financialpost*. https://financialpost.com/personal-finance/mortgages-real-estate/now-and-then-do-canadian-homes-really-cost-that-much-more-than-30-years-ago

[2] Statistics Canada, “Income of individuals by age group, sex and income source, Canada, provinces and selected census metropolitan areas,” *Statcan.gc.ca*, Apr. 26, 2024. https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1110023901

[3] “Toronto Median Price | CREA Statistics,” *creastats.crea.ca*. https://creastats.crea.ca/mls/treb-median-price

[4] “S&P 500 (^GSPC) Charts, Data & News - Yahoo Finance,” *finance.yahoo.com*. https://finance.yahoo.com/quote/%5EGSPC

[5] *Realtor.ca*, 2025. https://www.realtor.ca/map#view=list&Sort=6-D&GeoIds=g30_dpz89rm7&GeoName=Toronto%2C%20ON&PropertyTypeGroupID=1&TransactionTypeId=2&PropertySearchTypeId=3&SQFTRange=700-800&OwnershipTypeGroupId=2&Currency=CAD.

[6] C. Rogers, “Canada’s mortgage market—A question of balance,” *Bankofcanada.ca*, Nov. 06, 2024. https://www.bankofcanada.ca/2024/11/canadas-mortgage-market-a-question-of-balance/

[7] XGBoost Developers, *XGBoost Documentation – stable version*, Available: https://xgboost.readthedocs.io/en/stable/

Appendix

Past References

M. Bostock, S. Carter, A. Tse, and F. Paris, “Is It Better to Rent or Buy? A Financial Calculator.,” *The New York Times*, May 10, 2024. Available: https://www.nytimes.com/interactive/2024/upshot/buy-rent-calculator.html

Government of Ontario, “Residential rent increases,” *Ontario.ca*, 2021. https://www.ontario.ca/page/residential-rent-increases

J. B. Maverick, “S&P 500 Average Return and Historical Performance,” *Investopedia*, Dec. 26, 2024. https://www.investopedia.com/ask/answers/042415/what-average-annual-return-sp-500.asp

“Housing Market Information Portal,” *CMHC*, 2024. https://www03.cmhc-schl.gc.ca/hmip-pimh/en/TableMapChart/Table?TableId=1.1.1.8&GeographyId=2270&GeographyTypeId=3&DisplayAs=Table&GeograghyName=Toronto#TableMapChart/2270100/6/Mount%20Pleasant%20West.

Statcan.gc.ca, 2020. https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1010013901

“Open Data Dataset.” https://open.toronto.ca/dataset/neighbourhood-profiles/

“Statistics,” *Bankofcanada.ca*, 2019. https://www.bankofcanada.ca/rates/

GitHub Repository

All project phases, iterations, and results will be available in the project repository at https://github.com/MatteoVDP/Buying-vs.-Renting-a-Property-in-Toronto.

Future Value Formula for Calculating Stock Market Returns

$$FV = PV(1 + \frac{r}{n})^{nt} + P_c \frac{((1 + \frac{r}{n})^{nt} - 1)}{(\frac{r}{n})}.$$

Where:
FV = Future Value of the investment.
PV = Present Value or the initial principal amount.
R = Annual interest rate.
N = The number of times the interest is compounded per year.
T = The time the money is invested for, in years.
Pc = The amount of the periodic contribution, added at the end of each compounding period.