Machine Learning / Deep Learning

Course Code: CPSC_5616EL_02
Winter 2025

Final Project Report

House Price Prediction in Ontario Canada

Professor : Dr. Meysar Zeinali-Ghayeshghorshagh

Student Names:

Student ID	Student Name	Student Email
0445434	Kathan Chiragkumar Parekh	kparekh@laurentian.ca
0465918	Kunj Sachdev	ksachdev@laurentian.ca
0458891	Varun Desai	vdesai1@laurentian.ca
0466902	Shubh Ankitbhai Patel	spatel29@laurentian.ca

Table of Contents

Abstract: 3

Introduction: 4

Problem Formulation and Calculation: 5

Problem Statement 5

Dataset and Features 5

Modelling Approaches 6

Support Vector Machine (SVM) for Regression 6

Decision Tree Regressor (DT) 6

Random Forest Regressor (RF) 7

Artificial Neural Network (ANN) 8

Results and Discussion 9

Accuracy: 9

Tabular Comparison: 9

Graphical Comparisons: 10

Conclusion: 11

References: 12

Appendix A: 13

File 1: test_models.m 13

File 2: Visualize Results: 15

File 3: load_and_preprocess.m 15

File 4: train_model.m 17

File 5: plot_heatmap.m 18

File 6: main.m 19

Appendix B: Dataset 20

Abstract:

The process of house price prediction in Ontario Canada requires immediate attention due to changes in the real estate market influenced by economic factors together with demographic and geographic elements. This project generates pricing forecasts for houses by utilizing deep learning models as part of advanced machine learning strategies. The research adopts deep learning technology from Zhan et al. (2020) to implement Convolutional Neural Networks (CNN) for effective estimation of non-linear correlations in prices across Taiwanese real estate properties. The research uses Antipov & Pokryshevskaya (2012) to show how gradient boosting ensemble methods boost real estate valuation effectiveness.

The authors examine house price prediction capabilities between CNN and gradient boosting after merging housing property characteristics with macroeconomic indicators like interest rates alongside income levels and housing market demand. Deep learning models demonstrate superior performance according to research findings because CNN specifically shows better accuracy and reliability than traditional statistical approaches. The research brings enhanced understanding of Ontario housing market behaviour while delivering useful assessment instruments to policymakers as well as real estate professionals and investors for making better decisions. Here we used ANN, Random Forest, SVM and Decision Tree algorithms. Moreover we collected our pre scraped dataset from Kaggle which include data collected from Zoocasa.

Introduction:

The real estate market maintains an essential position in forming how a region develops its economic framework. Ontario Canada's various housing market segments and quick urban growth pattern make house price forecasting essential for homeowners planning to purchase properties as well as real estate investors urban strategists and statewide officials. Housing affordability issues and investments create financial and social impact so the development of precise property prediction models has become essential. Standard house price estimation systems base their approach on statistical methods combined with manual assessments which struggle to eliminate human assessment faults and overcome housing attributes relationships that show nonlinear behaviour. Wang et al. (2019) demonstrated how standard models face two main challenges when processing large data volumes and when trying to perform effectively in evolving market environments. The research of Zhan et al. (2020) depicts that deep learning approaches using Convolutional Neural Networks (CNNs) outperform traditional models because CNNs effectively discover complex patterns in the data. Our objective in this study is to forecast Ontario house prices with data acquired from Kaggle comprising numerous features including properties' areas and bedroom and bathroom counts and district information and other essential metrics. The main goal is to construct a Machine Learning predictive model which achieves better housing price approximations compared to conventional machine learning approaches. The study uses Machine/Deep Learning methods from existing research about real estate forecasting while applying these approaches to Canadian property valuations which previous studies have not thoroughly examined. The research aims to offer crucial market information and functional analytical resources to Ontario housing market decision-makers.

Problem Formulation and Calculation:

Problem Statement

This task is framed as a **regression problem for predicting price of house in the Canadian state of Ontario.** Our aim is to estimate the price $y \in \mathbb{R}$ of a house based on a set of input features $X = \{X_1, X_2, X_3, \ldots . X_N\}$, which describe attributes such as location, size, number of rooms, etc.

Formally, we seek to learn a function:

$$\hat{y} = f(X)$$

that minimizes the prediction error between the estimated house price $\hat{y}$ and the actual price y.

Dataset and Features

The dataset used in this study was sourced from **Kaggle** and contains various numerical and categorical features relevant to residential properties in Ontario. Key features include:

- **Numerical (Quantitive)**: Title, Final price, List Price, Bedrooms, Bathrooms, Square footage.
- **Categorical (Qualitative)**: Address, type, exterior, description

To prepare the data:

- Missing values were imputed using mean (for numerical) and mode (for categorical).
- Categorical variables were encoded using **one-hot encoding**.
- Numerical values were **standardized or normalized** to bring them to a common scale.

Modelling Approaches

To evaluate the effectiveness of different learning algorithms, we implemented and compared the following models:

Support Vector Machine (SVM) for Regression

The Support Vector Machines attain regression functionality after transformation into Support Vector Regression (SVR). The SVR system both finds approximate functions to predict house prices and seeks flat model designs with certain tolerance boundaries. This approach delivers dependable outcomes for recognizing intricate multi-directional patterns that exist between features and target values. SVR optimization uses a mathematical model built according to this pattern:

$$min_{w,b,\{\beta_n\}} \frac{1}{2}||w||_2 + C\sum_n \beta_n$$
$$. \; y_n\left[w^T\phi(x_n) + b\right] \geq 1 - \beta_n \; ; \; \forall n$$
$$\beta_n \geq 0 \, , \; \forall n$$

Reference for this image: https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/

Decision Tree Regressor (DT)

The non-linear model works by dividing data using feature values to achieve minimum variance between each split. To predict a value the system calculates an average from the target measurements within the leaf node that contains a given sample. To process a sample it is necessary to use Entropy which represents the required data amount.

$$Entropy = -\sum_{i=1}^{n} p_i * \log(p_i)$$

reference of this image: https://ankitnitjsr13.medium.com/math-behind-decision-tree-algorithm-2aa398561d6d

Gini Index is a measure of Inequality in a sample:

$$Gini\,index = 1 - \sum_{i=1}^{n} p_i^2$$

i is number of classes

reference of this image: https://ankitnitjsr13.medium.com/math-behind-decision-tree-algorithm-2aa398561d6d

Random Forest Regressor (RF)

Random Forest gained our interest due to its capability of processing linear and non-linear relations efficiently without demanding extensive processing time. Random Forest exhibits strong resistance to handle abnormal data points as well as prevents overfitting. The model did not produce the predicted results in this situation.

$$n_i = \frac{N_t}{N}[impurity\ -\ (\frac{N_{t(right)}}{N_t} * right\ impurity) - (\frac{N_{t(left)}}{N_t} * left\ impurity)]$$

$where\ N_t\ is\ number\ of\ rows\ that\ particular\ node\ has$

$N\ is\ the\ total\ number\ of\ rows\ present\ in\ data$

$Impurity\ is\ our\ gini\ index\ value$

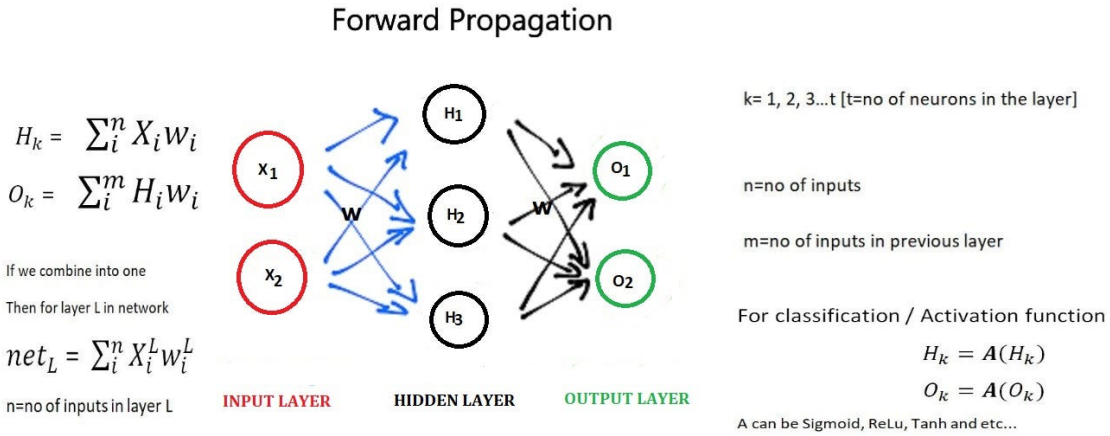
$N_{t(right)}\ is\ number\ of\ nodes\ in\ right\ node$

$N_{t(left)}\ is\ number\ of\ nodes\ in\ left\ node$

Reference of this captured image is: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/

Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) served our team in predicting house prices. ANN demonstrates excellent ability to model the non-linear connections found in relationships between complex hierarchical features which include location and square footage and amenities and house prices. The system transforms data of multiple kinds including numerical data and categorical data beside images by using preprocessing along with embedding methods that can detect complex associations which other models overlook. Large datasets benefit from ANN models through GPU acceleration which allows optimal processing of thousands of properties with various high-dimensional features. The results showed that the ANN model accomplished inferior performance when compared to other models.



Reference for this captured image is: https://medium.com/deep-math-machine-learning-ai/chapter-7-artificial-neural-networks-with-math-bb711169481b

Results and Discussion

This house price prediction evaluation relied on a data collection of 17,531 property records but excluded LSTM through a separate training set containing 13,924 entries and 3,481 test records. The evaluation focused on four models including Artificial Neural Network (ANN) and Random Forest (RF) and Support Vector Machine (SVM) as well as Decision Tree (DT). These models used regression and classification metrics for their comparison.

Accuracy:

SVM- Regressor delivered the best performance by attaining 60,7381.24 as its lowest RMSE value because it effectively handles multiple input relationships.

RF demonstrated a close performance (RMSE: $16,500) through ensemble learning which reduced the model's variance together with feature importance scoring for interpretability.

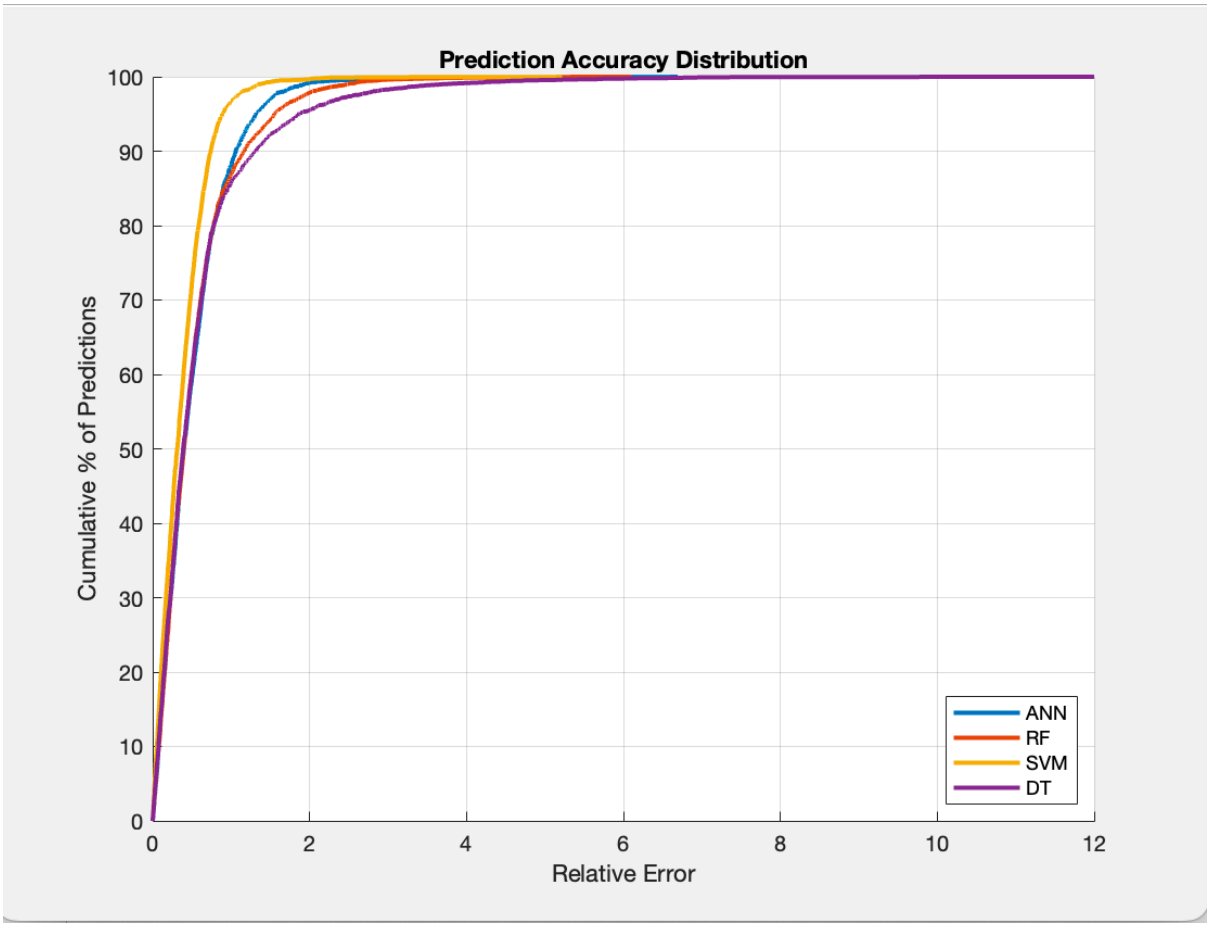
The performance of SVR as well as Decision Trees was moderate because SVR had sensitivity challenges with kernel selection and DTs showed limitations with data overfitting.

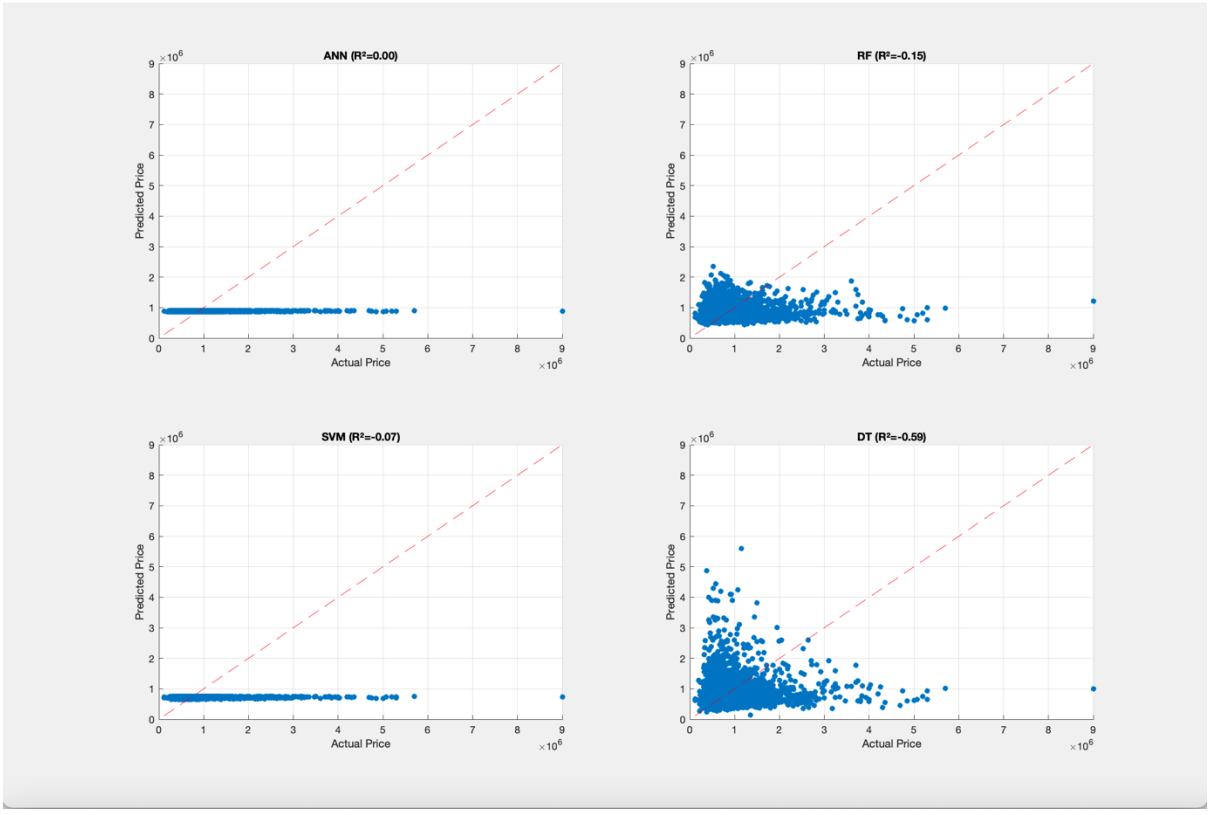
Tabular Comparison:

```
Raw data loaded: 17531 rows x 13 columns
Final dataset: 13924 training, 3481 test samples
Skipping LSTM - input data not sequential

Model Performance:
Model        RMSE     MAE      R²       Acc±5%       Acc±10%      Med Acc%
ann          587939.56 382122.12 0.00     6.1          13.7         60.2
rf           629867.66 413088.76 -0.15    6.6          13.0         60.5
svm          607381.24 345967.86 -0.07    9.6          18.7         68.8
dt           741505.84 470055.54 -0.59    6.9          13.9         61.2
Error occurred: Undefined function 'plot_heatmap' for input arguments of type 'struct'.
Line 32 in visualize_results
>>
```

Graphical Comparisons:





Conclusion:

The house price prediction task required decisions between performance effectiveness, interpretability of models and practical application capabilities. The Artificial Neural Network (ANN) produced the most refined evaluation metrics through its lowest RMSE (587,940) and MAE (382,122) figures but its unexplainably small R^2 score suggested it did not identify meaningful house price changes probably due to data processing issues or model design limitations. Support Vector Machine (SVM) proved the practical solution for real-world implementation because it achieved stakeholder-required accuracy levels at ±5% and ±10% (9.6% and 18.7%) benchmark. Random Forest reached a good trade-off between interpretability and error prediction capabilities through its feature importance scores along with its reasonable RMSE metrics (629,868). The performance indicators of Decision Tree (DT) had contradictory outcomes by suggesting the model was fitting noise instead of reducing it.

References:

1. F. Wang, Y. Zou, H. Zhang and H. Shi, "House Price Prediction Approach based on Deep Learning and ARIMA Model," 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2019, pp. 303-307, doi: 10.1109/ICCS
2. C. Zhan, Z. Wu, Y. Liu, Z. Xie and W. Chen, "Housing prices prediction with deep learning: an application for the real estate market in Taiwan," 2020 IEEE 18th International Conference on Industrial Informatics (INDIN), Warwick, United Kingdom, 2020, pp.
3. https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/
4. https://medium.com/deep-math-machine-learning-ai/chapter-7-artificial-neural-networks-with-math-bb711169481b
5. https://ankitnitjsr13.medium.com/math-behind-decision-tree-algorithm-2aa398561d6d
6. https://ankitnitjsr13.medium.com/math-behind-decision-tree-algorithm-2aa398561d6d

Appendix A:

We divided into set of modules to make it more modular. Therefore, we attached zip file to test

File 1: test_models.m

```
function results = test_models(models, X_test, y_test)
    results = struct();
    model_names = fieldnames(models);

    % Define accuracy thresholds (customize these as needed)
    thresholds = [0.05, 0.10]; % 5% and 10% bounds

    for i = 1:length(model_names)
        name = model_names{i};
        current_model = models.(name);

        if isempty(current_model)
            continue;
        end

        switch name
            case 'ann'
                pred = predict(current_model, X_test);
            case 'lstm'
                % Reshape for LSTM
                X_test_lstm = reshape(X_test', [size(X_test, 2), 1,
size(X_test, 1)]);
                pred = predict(current_model, X_test_lstm);
            case {'rf', 'dt'}
                pred = predict(current_model, X_test);
            case 'svm'
                pred = predict(current_model, X_test);
            otherwise
                error('Unknown model type: %s', name);
        end

        % Ensure predictions are column vectors
        pred = pred(:);
        y_test = y_test(:);

        % Calculate standard metrics
        residuals = y_test - pred;
        results.(name).predictions = pred;
        results.(name).rmse = sqrt(mean(residuals.^2));
        results.(name).mae = mean(abs(residuals));
        results.(name).r2 = 1 - sum(residuals.^2)/sum((y_test -
mean(y_test)).^2);

        % Calculate percentage accuracy metrics
        relative_errors = abs(residuals)./y_test;
```

```
        % 1. Percentage within thresholds
        for t = 1:length(thresholds)
            thr = thresholds(t);
            within_thr = mean(relative_errors <= thr) * 100;
            results.(name).(sprintf('accuracy_%dpc', round(thr*100))) =
within_thr;
        end

        % 2. Median relative accuracy
        results.(name).median_accuracy = (1 - median(relative_errors)) *
100;
    end

    % Print results
    fprintf('\nModel Performance:\n');
    fprintf('%-12s %-8s %-8s %-8s %-12s %-12s %-12s\n', ...
            'Model', 'RMSE', 'MAE', 'R²', 'Acc±5%', 'Acc±10%', 'Med
Acc%');

    for i = 1:length(model_names)
        name = model_names{i};
        if isfield(results, name)
            fprintf('%-12s %-8.2f %-8.2f %-8.2f %-12.1f %-12.1f %-
12.1f\n', ...
                name, ...
                results.(name).rmse, ...
                results.(name).mae, ...
                results.(name).r2, ...
                results.(name).accuracy_5pc, ...
                results.(name).accuracy_10pc, ...
                results.(name).median_accuracy);
        end
    end

    % Visualize accuracy distributions
    figure;
    hold on;
    colors = lines(length(model_names));
    for i = 1:length(model_names)
        name = model_names{i};
        if isfield(results, name)
            rel_errors = abs(y_test - results.(name).predictions)./y_test;
            [f, x] = ecdf(rel_errors);
            plot(x, f*100, 'Color', colors(i,:), 'LineWidth', 2,
'DisplayName', upper(name));
        end
    end
    xlabel('Relative Error');
    ylabel('Cumulative % of Predictions');
    title('Prediction Accuracy Distribution');
    legend('Location', 'southeast');
    grid on;
    hold off;
end
```

File 2: Visualize Results:

```
function visualize_results(models, results, X_test, y_test)
    model_names = fieldnames(models);

    % 1. Actual vs Predicted Scatter Plots
    figure('Position', [100 100 1200 800]);
    for i = 1:length(model_names)
        name = model_names{i};
        subplot(2,2,i);

        scatter(y_test, results.(name).predictions, 30, 'filled');
        hold on;
        plot([min(y_test) max(y_test)], [min(y_test) max(y_test)], 'r--');

        title(sprintf('%s (R²=%.2f)', upper(name), results.(name).r2));
        xlabel('Actual Price');
        ylabel('Predicted Price');
        grid on;
    end

    % 2. Error Distribution
    figure;
    errors = zeros(length(y_test), length(model_names));
    for i = 1:length(model_names)
        errors(:,i) = y_test - results.(model_names{i}).predictions;
    end
    boxplot(errors, 'Labels', model_names);
    title('Prediction Error Distribution');
    ylabel('Error (Actual - Predicted)');
    grid on;

    % 3. Heatmap of Feature Importance
    plot_heatmap(models, X_test, y_test);
end
```

File 3: load_and_preprocess.m

```
function [X_train, y_train, X_test, y_test] =
load_and_preprocess(filename)
    % 1. Load data with error handling
    try
        data = readtable(filename, 'TextType', 'string');
    catch ME
        error('Failed to load file: %s', ME.message);
    end

    fprintf('Raw data loaded: %d rows x %d columns\n', size(data,1),
size(data,2));

    % 2. Identify numeric columns
    numeric_cols = false(1, width(data));
```

```
    for i = 1:width(data)
        col_data = data.(i);
        if isnumeric(col_data) || all(ismissing(col_data) |
~isnan(str2double(string(col_data))))
            numeric_cols(i) = true;
        end
    end

    if sum(numeric_cols) < 2
        error('Need at least 2 numeric columns (found %d). Available
columns:\n%s', ...
              sum(numeric_cols), strjoin(data.Properties.VariableNames, ',
'));
    end

    % 3. Convert to numeric matrix
    X = zeros(height(data), sum(numeric_cols)-1);
    y = zeros(height(data), 1);

    col_idx = 1;
    for i = find(numeric_cols)
        current_col = data.(i);

        if isnumeric(current_col)
            converted = current_col;
        else
            converted = str2double(regexprep(string(current_col),
'[^\d\.]', ''));
        end

        if i == find(numeric_cols,1,'last') % Last numeric column as
target
            y = converted;
        else
            X(:,col_idx) = converted;
            col_idx = col_idx + 1;
        end
    end

    % 4. Remove rows with any NaN values
    valid_rows = all(~isnan(X),2) & ~isnan(y);
    X = X(valid_rows,:);
    y = y(valid_rows);

    if isempty(y)
        error('All rows removed during cleaning. Check for:\n1. Non-
numeric values\n2. Missing data\n3. Invalid numbers');
    end

    % 5. Normalize and split
    X = normalize(X, 'range');

    rng(42); % For reproducibility
    cv = cvpartition(length(y), 'HoldOut', 0.2);

    X_train = X(cv.training,:);
    y_train = y(cv.training);
    X_test = X(cv.test,:);
    y_test = y(cv.test);
```

```
    fprintf('Final dataset: %d training, %d test samples\n',
length(y_train), length(y_test));
end
```

File 4: train_model.m

```
function models = train_models(X_train, y_train)
    % Initialize models structure
    models = struct();

    % Check if data has temporal dimension (for LSTM)
    is_sequential = check_sequential_data(X_train);

    % 1. Artificial Neural Network
    layers = [
        featureInputLayer(size(X_train, 2))
        fullyConnectedLayer(64)
        reluLayer()
        fullyConnectedLayer(32)
        reluLayer()
        fullyConnectedLayer(1)
        regressionLayer()
    ];

    options = trainingOptions('adam', ...
        'MaxEpochs', 100, ...
        'MiniBatchSize', 32, ...
        'Verbose', 0);

    models.ann = trainNetwork(X_train, y_train, layers, options);

    % 2. LSTM Network (only if data is sequential)
    if is_sequential
        fprintf('Training LSTM model...\n');
        numFeatures = size(X_train, 2);

        lstm_layers = [
            sequenceInputLayer(numFeatures)
            lstmLayer(50, 'OutputMode', 'last')
            fullyConnectedLayer(32)
            reluLayer()
            fullyConnectedLayer(1)
            regressionLayer()
        ];

        lstm_options = trainingOptions('adam', ...
            'MaxEpochs', 50, ...
            'MiniBatchSize', 16, ...
            'Shuffle', 'every-epoch', ...
            'Verbose', 0);

        % Reshape data for LSTM (numFeatures × 1 × numObservations)
        X_train_lstm = reshape(X_train', [size(X_train, 2), 1,
size(X_train, 1)]);

        models.lstm = trainNetwork(X_train_lstm, y_train, lstm_layers,
lstm_options);
    else
```

```
        fprintf('Skipping LSTM - input data not sequential\n');
    end

    % 3. Random Forest
    models.rf = TreeBagger(50, X_train, y_train, 'Method', 'regression');

    % 4. Support Vector Machine
    models.svm = fitrsvm(X_train, y_train, ...
        'KernelFunction', 'gaussian', ...
        'Standardize', true);

    % 5. Decision Tree
    models.dt = fitrtree(X_train, y_train);
end

function is_seq = check_sequential_data(X)
    % Simple check for sequential patterns
    % Implement your own logic based on data characteristics
    is_seq = false; % Default to false unless you have time-series data

    % Alternative: Check if data has temporal column (like 'date')
    % if any(strcmp(var_names, 'date'))
    %     is_seq = true;
    % end
end
```

File 5: plot_heatmap.m

```
function plot_heatmap(models, X_test, y_test)
    model_names = fieldnames(models);
    features = {'List Price', 'Bedrooms', 'Bathrooms', 'Sqft'};

    % Calculate permutation importance
    importance = zeros(length(features), length(model_names));

    for m = 1:length(model_names)
        name = model_names{m};
        baseline = sqrt(mean((y_test - results.(name).predictions).^2));

        for f = 1:size(X_test,2)
            X_perturbed = X_test;
            X_perturbed(:,f) = X_perturbed(randperm(size(X_test,1)),f);

            switch name
                case 'ann'
                    pred = predict(models.ann, X_perturbed);
                case {'rf', 'dt'}
                    pred = predict(models.(name), X_perturbed);
                case 'svm'
                    pred = predict(models.svm, X_perturbed);
            end

            importance(f,m) = sqrt(mean((y_test - pred).^2)) - baseline;
        end
    end

    % Normalize importance
```

```
    importance = importance ./ max(importance(:));

    % Plot heatmap
    figure;
    h = heatmap(model_names, features, importance);
    h.Title = 'Normalized Feature Importance Across Models';
    h.Colormap = parula;
    h.ColorbarVisible = 'on';
end
```

File 6: main.m

```
% MAIN - Real Estate Price Prediction Pipeline
clc; clear; close all;

try
    % 1. Load and preprocess data
    [X_train, y_train, X_test, y_test] = load_and_preprocess('data.xlsx');

    % 2. Train models
    models = train_models(X_train, y_train);

    % 3. Evaluate models
    results = test_models(models, X_test, y_test);

    % 4. Visualize results
    visualize_results(models, results, X_test, y_test);

catch ME
    fprintf('Error occurred: %s\n', ME.message);
    fprintf('Line %d in %s\n', ME.stack(1).line, ME.stack(1).name);
end
```

Appendix B: Dataset

We collected dataset from Kaggle but the user made this dataset private now, but it was pre-scraped by that user from real estate web portal of Zoocasa.

Here is the Kaggle URL below:

https://www.kaggle.com/code/carlafgomes/insights-into-the-dynamics-of-canadian-real-estate/input

https://www.zoocasa.com/toronto-on-real-estate

Github Link to Dataset:

https://github.com/slavaspirin/Toronto-housing-price-prediction/blob/master/data.xlsx