**A red and black logo

Description automatically generated**

**Final Project: Multi-task Learning for Predicting House Prices and House Category**

Presented to

Professor Fatih Nayebi

By

Barabasz, Michelle - 261152119

MGSC 673

McGill University - Desautels Faculty of Management

**Table of Contents**

[*1. Introduction* 3](#_Toc165234329)

[*2. Data Exploration and Preprocessing* 3](#_Toc165234330)

[*2.1. Handling Missing Values* 3](#_Toc165234331)

[*2.2. Encoding and Normalization of Variables* 3](#_Toc165234332)

[*2.3. Removal of Redundant or Leaky Variables* 4](#_Toc165234333)

[*2.4. Splitting the Data* 4](#_Toc165234334)

[*2.5. Defining the House Category* 4](#_Toc165234335)

[*3. Multi-task Model Building* 4](#_Toc165234336)

[*3.1. Architectural Design* 4](#_Toc165234337)

[*3.2. Branching into Task-specific Layers* 5](#_Toc165234338)

[*3.3. Utilizing PyTorch Lightning* 5](#_Toc165234339)

[*3.4. Activation Functions and Optimizers* 5](#_Toc165234340)

[*3.5. Advanced PyTorch Lightning Features* 6](#_Toc165234341)

[*3.6. Loss Functions* 6](#_Toc165234342)

[*3.7. Hyperparameter Optimization* 6](#_Toc165234343)

[*3.8. Validation Methodology Extension* 7](#_Toc165234344)

[*3.9. Experimentation Results* 7](#_Toc165234345)

[*4. Final Model Results* 7](#_Toc165234346)

[*5. Conclusion* 8](#_Toc165234347)

# *1. Introduction*

This project aims to utilize machine learning’s innovative approaches by developing a multi-task neural network tailored to the real estate market. The objective has two parts: to predict house prices and to categorize houses into distinct categories. To achieve this, a comprehensive machine learning pipeline was created which integrates data exploration with the robust model-building capabilities of PyTorch Lightning.

The project uses the advanced features offered by PyTorch Lightning, including its intuitive logging, versatile callback system, and the sophisticated Trainer API, providing a streamlined way to manage the intricacies of the multi-task learning model. Alongside these tools, Optuna will be used for hyperparameter optimization, ensuring that the model is not only effective but also finely tuned for optimal performance. By completing these steps, a model that stands at the intersection of innovation and functionality, adept at interpreting the complex patterns of the housing market to produce valuable predictions and classifications will be delivered.

# *2. Data Exploration and Preprocessing*

The data exploration and preprocessing stage was crucial in preparing the dataset for the multi-task learning model. It began by conducting a comprehensive data exploration, which included a statistical summary of the features, and a visual examination of numerical variable distributions and their relationships with target variable, Sale Price. This allowed for uncovering underlying patterns, detect outliers, and grasp the data's overall structure.

## *2.1. Handling Missing Values*

In handling missing values, strategies that best fit the nature of each variable were adopted. For categorical variables, the label 'None' was assigned to account for missing data, treating it as a distinct category. This approach is particularly useful when the absence of information is informative in itself. Numerical variables, on the other hand, were filled with their median values. The median is a robust measure that is not skewed by outliers, making it a good choice for maintaining the integrity of the data's distribution.

## *2.2. Encoding and Normalization of Variables*

Moving to encoding categorical variables, OneHotEncoder was utilized, which converted each category into a new binary feature. This was done for all categorical features, ensuring they were appropriately formatted for model input.

Normalization of numerical variables was also an essential step in the preprocessing pipeline. RobustScaler was employed, which is a technique that is less influenced by outliers compared to other scaling methods like Min-Max scaling or Standard Scaling. By using the median and interquartile range, RobustScaler provided a scaling mechanism that respects the presence of outliers, preventing the model from being unduly affected by extreme value variations.

## *2.3. Removal of Redundant or Leaky Variables*

In the preprocessing efforts, each variable was carefully evaluated for its relevance and potential to introduce data leakage. It was decided to remove variables that could inadvertently provide the model with future information or were redundant due to correlation with other features. ‘MSSubClass’ was removed to prevent data leakage as it appeared to have similar information to the target variables related to house type. Additionally, ‘Id’ was omitted as it serves no predictive purpose. Features directly related to the outcome, like ‘SalePrice’, were reserved for training targets only and not used as inputs to the model.

## *2.4. Splitting the Data*

From the dataset, 80% of the observations (1,168 observations) were dedicated to the training set, while the remaining 20% of observations (292 observations) were dedicated to the test set.

## *2.5. Defining the House Category*

The creation of the new House Category variable was dependent on four pre-existing variables: House Style, Buidling Type, Year Built, and Year Remodeled/Added. To create this new category, a Python function was defined to assign categories using a set of conditions: properties built before 1950 and not remodeled since were classified as ‘Historic’, while those built or remodeled after 1950 were deemed ‘Modern’. The house size, inferred from the number of floors, led to labels such as ‘Two-Story’ or ‘One-Story’. Architectural styles were derived from the building type and house style, resulting in categories like ‘Single-Family One-Story’ or ‘Single-Family Two-Story’. Additionally, properties were categorized based on their construction era and locality, distinguishing between ‘Old Neighborhood’ and ‘New Neighborhood’. Houses that did not fit any specific criteria were thoughtfully classified as ‘Other’, ensuring a comprehensive categorization of the dataset.

# *3. Multi-task Model Building*

At the heart of this project was the construction of a sophisticated multi-task neural network designed to simultaneously address regression and classification challenges: predicting house prices and categorizing house categories.

## *3.1. Architectural Design*

The multi-task model was architected with a shared foundation—common layers that both task-specific components could build upon. This shared bottom architecture aimed to capture rich, generalizable representations applicable to both tasks. The hypothesis was that certain features relevant to price prediction would also provide valuable insights into categorical classification, thereby allowing the model to leverage shared knowledge and reduce overall complexity.

## *3.2. Branching into Task-specific Layers*

After the shared layers, the network diverged into two distinct pathways, each fine-tuned for its specific objective:

* Regression Head: This branch of the network was crafted with layers adept at outputting a continuous value indicative of the house's market price. The design was optimized to understand the nuances of numerical data and accurately forecast a house’s value.
* Classification Head: This branch was composed of layers specialized in categorical discernment. These layers were engineered to process the shared features and classify the input into discrete house type categories with precision.

## *3.3. Utilizing PyTorch Lightning*

To orchestrate the complexities of building and managing such a multi-faceted model, the PyTorch Lightning framework was utilized. PyTorch Lightning allowed for abstracting away the boilerplate code typically associated with deep learning models, allowing for further focus on the novel aspects of the architecture. Through this framework, the model's structure could be defined, the data loading process streamlined, and the training loop managed with greater ease and reproducibility. This not only facilitated faster experimentation but also brought structure and clarity to the implementation of the multi-task learning solution.

The building of the multi-task model was a meticulous process of balancing the intricacies of shared and task-specific representations. The model had to be capable of generalizing across tasks while still preserving the specialized knowledge required for accurate price regression and categorical classification. Through the combination of thoughtful architectural design and the robust capabilities of PyTorch Lightning, a model was created to deliver insightful predictions on both fronts of the housing market's dynamics.

## *3.4. Activation Functions and Optimizers*

A variety of activation functions and optimizers, key components that significantly influence model performance, were explored. Activation functions like ReLU and Leaky ReLU were trialed, each introducing non-linearity to the learning process, while the effects of different optimizers—including Adam—were examined. The ReLU function was primarily chosen for its ability to maintain gradient flow and reduce the likelihood of vanishing gradients, an essential consideration for deep networks. Conversely, Leaky ReLU was tested to mitigate the “dying ReLU” problem by allowing a small, non-zero gradient when the unit is not active.

Optimizers play a crucial role in navigating the loss landscape. The choice of Adam was dictated by its adaptive learning rate capabilities, which are particularly effective in handling the sparse gradients and varying data scales that are characteristic of housing data. These components were iteratively assessed for their impact on convergence speed and model accuracy, forming a vital part of the network’s architecture refinement.

## *3.5. Advanced PyTorch Lightning Features*

In the development of the multi-task neural network for predicting house prices and categorizing houses, the project leveraged advanced features of PyTorch Lightning to streamline and enhance the model building and training process. The use of PyTorch Lightning's logging feature allowed for meticulous recording of model performance metrics during training, which facilitated in-depth analysis and fine-tuning. The callback system proved indispensable, automating essential tasks such as model checkpointing, early stopping, and learning rate adjustments. These callbacks were configured to monitor training progress and make adjustments on-the-fly, ensuring that the model trained efficiently and effectively without overfitting or undertraining.

The Trainer API provided a high-level interface for the training loop, abstracting away the complexities of manually coding the training iterations. This allowed for a more focused effort on model architecture and hyperparameter tuning, rather than boilerplate code. Through the Trainer API, the project could easily scale the model training to different hardware setups without changing the model code, further attesting to the flexibility and efficiency of PyTorch Lightning in managing complex machine learning projects.

These advanced features contributed significantly to the project's success, enabling the multi-task neural network to learn from the data effectively and providing the team with a robust platform for experimentation and iteration.

## *3.6. Loss Functions*

For this project, a compound loss function that synergizes the loss for regression (Mean Squared Error) and classification tasks (Cross-Entropy) was used. This integrated loss function provided a unified measure of performance across both tasks, essential for the simultaneous training of the multi-task model. The weighted sum of the MSE for price prediction and Cross-Entropy for house category classification was carefully balanced to ensure neither task dominated the learning process, facilitating a cohesive training regimen. Specifically, the losses were weighted equally in the initial configuration, with both the MSE and Cross-Entropy losses assigned a weight of 0.5.

## *3.7. Hyperparameter Optimization*

The multi-task neural network underwent extensive hyperparameter tuning through Optuna, a powerful and flexible hyperparameter optimization framework. This process was crucial to fine-tune various aspects of the model architecture and training regimen to achieve the best performance. The specific hyperparameters explored included:

* Number of Hidden Layers: The architecture was varied to have between 1 to 3 hidden layers to determine the optimal depth of the network for handling the complexity of the task.
* Hidden Layer Size: Each hidden layer's size was adjusted, with options ranging from 30 to 500 neurons. This allowed the model to explore different capacities for feature representation.
* Dropout Rate: To prevent overfitting, dropout rates between 0.0001 and 0.9 were tested. Dropout helps in making the model robust by randomly deactivating a fraction of neurons during training.
* Learning Rate: This was varied logarithmically from 0.0001 to 0.1 to find the optimal speed at which the model learns. The right learning rate can significantly influence the convergence and stability of the training process.

## *3.8. Validation Methodology Extension*

To ensure the robustness of the multi-task learning model, a rigorous evaluation strategy was conducted. Although the data was divided into training and test subsets, k-fold cross-validation was also utilized during the hyperparameter tuning phase. This technique provided a comprehensive assessment of the model's generalization capabilities across different data segments, thus reinforcing the credibility of the performance metrics presented.

## *3.9. Experimentation Results*

The extensive hyperparameter tuning undertaken during the experimentation phase resulted in the identification of an optimal set of parameters. The Optuna framework's methodical search determined the best hyperparameters to be:

* Learning Rate (lr): 0.09
* Hidden Layer Size: 230 neurons
* Dropout Rate: 0.16

# *4. Final Model Results*

Utilizing the full potential of the training data, the final iteration of the multi-task model was subjected to an extensive training regimen, encompassing 1000 epochs. The validation loss recorded was 3.394, reflecting the model's ability to generalize from the training data.

The following metrics were found for the regression component:

* Sale Price RMSE: The model achieved a Root Mean Square Error (RMSE) of $36,813.64, a measure of the average deviation of the price predictions from the actual values.
* Mean Absolute Error (MAE): With an MAE of $21,454.85, the model on average deviated from the true sale prices by this amount, providing a clear perspective on the model's accuracy in dollar terms.
* R2 Score: An R2 score of 0.82 indicated that the model’s predictions closely followed the variance of the true sale prices, suggesting a high level of explanatory power.

When evaluating the classification component, the metrics were found:

* Accuracy: The accuracy of the model stood at 79.11%, denoting a high level of correctness in the categorical predictions when compared to the baseline heuristics.
* Precision: At 62.58%, the precision metric showed a moderate level of exactness in the model's predictions, reflecting the proportion of true positive identifications among all positive classifications.
* Recall: Mirroring the accuracy, a recall of 79.11% indicated that the model was proficient at identifying the majority of positive class samples.
* F1-score: With an F1-score of 69.88%, the model demonstrated a balanced relationship between precision and recall, signifying a harmonic mean of the two metrics.

The metrics show that the model strikes a harmonious balance between predicting continuous numerical variables and classifying categorical data.

# *5. Conclusion*

In conclusion, this project not only showcases the power of a multi-task neural network in predicting house prices and categorizing houses but also demonstrates the versatility of PyTorch Lightning in simplifying complex model management. Through meticulous data preprocessing, thoughtful model architecture design, and rigorous hyperparameter optimization with Optuna, a model that adeptly handles both regression and classification tasks was developed. The final model's evaluation metrics attest to its robust predictive capabilities, reflecting its proficiency in capturing the intricate dynamics of the real estate market. With accuracy rates and RMSE values that substantially outperform baseline heuristics, the outcomes of this study reinforce the efficacy of the chosen methodologies and suggest a high potential for practical applications. Future explorations may delve into refining the model further, potentially incorporating additional data sources or exploring more sophisticated ensemble methods, ultimately enhancing the model's precision and broadening its applicability in the ever-evolving domain of real estate analytics.