House Price Predictor

Presented by:

Emerson Zahab, Jorge Santos Jr,
Anton LeBeque & Janelle Niznik

Solving, Analyzing, and Visualizing
Housing Price Prediction through
Machine Learning Modeling

Project Overview

Purpose

Our main aim is to leverage our skills in machine learning modeling to make reliable predictions on housing prices in Boston.

Approach

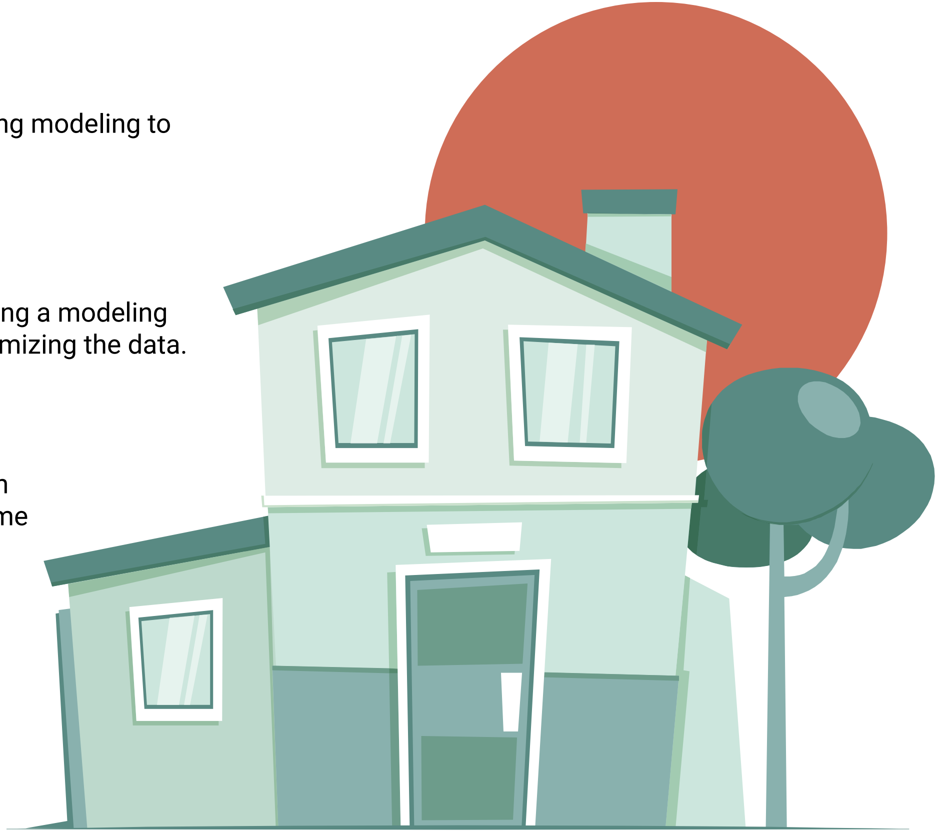
The proposed plan includes data acquisition, establishing a modeling process, conducting data modeling, and ultimately optimizing the data.

Rationale

Integrating machine learning in housing price prediction offers accuracy, informed decision-making, cost and time savings, market insights, scalability, adaptability, and reduced bias for the real estate industry.

Expected Outcome

This analysis leverages historical trends and predictive features to generate estimates of future home prices.



Plan of Action

Our approach to problem solving using Machine Learning

Acquire Dataset	Kaggle	Download CSV from Kaggle and retrieve using pyspark and tensorflow
Process for Modeling	Exploring the Dataset	Clean and process the dataset for modeling
Modeling	sklearn/tensorflow/matplotlib	Use machine learning modeling methods to make predictions
Optimization	hyperparameters/training/validation	Re-evaluate model(s) for better accuracy

The Boston Housing Data

Source

Derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA

Count

Number of rows: 394
Number of columns: 14

Warning

Ethical problem: the authors of this dataset included a variable that may appear to assume that racial self-segregation influences house prices

https://www.kaggle.com/datasets/altavish/boston-housing-dataset?resource=download&select=HousingData.csv

Data Processing

```
# Read the CSV file
df1 = spark.read.csv("../Resources/HousingData.csv", header=True, inferSchema=True)

# Show the content of the DataFrame
df1.show()
```

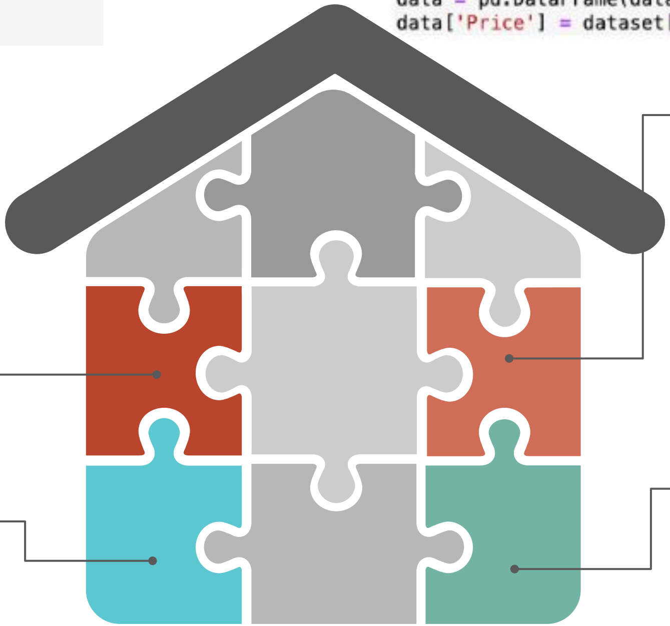
```
column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RA
data = pd.DataFrame(dataset[0][0], columns=column_names)
data['Price'] = dataset[0][1]
```

Read in Data

```
from tensorflow.keras.datasets import boston_housing
```

```
dataset = tf.keras.datasets.boston_housing.load_data()
```

```
# Load the dataset from a CSV file
data = pd.read_csv("../Resources/HousingData.csv")
```



Explore the data

```
# Determine the number of unique values in each column.
unique_counts = data.nunique()
```

```
# Check for missing values
train_df.isnull().sum()
```

Build dataframe

```
# Convert Spark DataFrame to Pandas DataFrame
pd_df = df.toPandas()
```

Clean the data

```
clean_data = data.dropna()
clean_data.head()
```

Evaluating & Cleaning the Data

In [3]:

```python
# Read the CSV file
df1 = spark.read.csv("../Resources/HousingData.csv", header=True, inferSchema=True)

# Show the content of the DataFrame
df1.show()
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24.0
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	NA	36.2
0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
0.08829	12.5	7.87	NA	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
0.17004	12.5	7.87	NA	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21.0	396.9	8.26	20.4
0.63796	0	8.14	NA	0.538	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2
0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9
1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21.0	386.85	6.58	23.1
0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21.0	386.75	14.67	17.5
0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21.0	288.99	11.69	20.2
0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21.0	390.95	11.28	18.2

only showing top 20 rows

```python
# Data Cleaning and Preprocessing
# Check for missing values
train_df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
target     0
dtype: int64
```

```python
# Check for duplicates
train_df.duplicated().sum()
```

```
0
```

Checked for: Missing, Duplicate and Null Values

Train-Test Split

The epoch is the # of times the model is going through this entire process, creating more variability each time its run or with a larger # of epochs

Full Dataset
Boston Housing Dataset

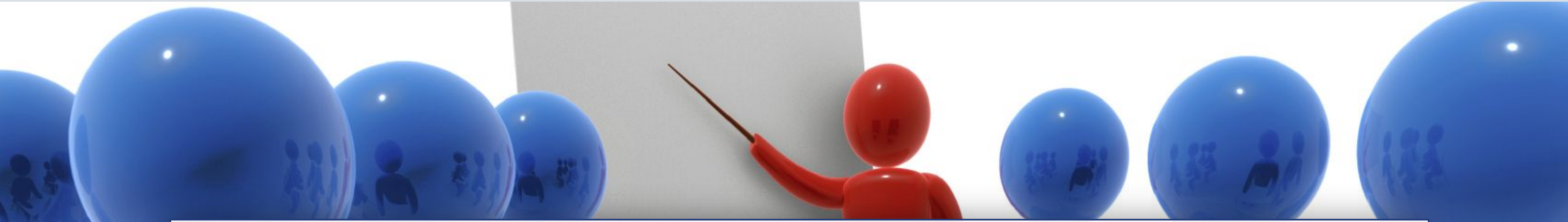
Training Set
80% for Training

Testing Set
20% for Testing

Training Set
80% of Training set for Training

Validation Set
20% of Training set for Validation

Train the Data



```
In [6]:
# split the data into 80% training and 20% test
X_train, X_test, y_train, y_test = train_test_split(
    train_data, train_targets, test_size=0.2, random_state=42)
```

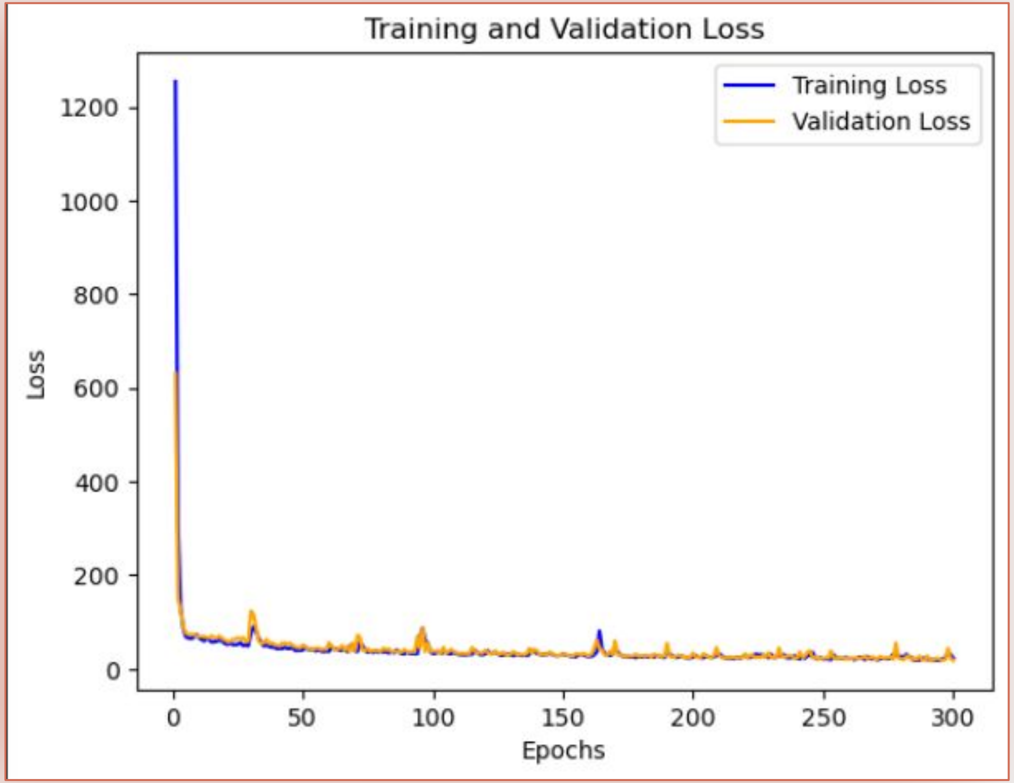
```
In [7]:
# print the shapes of the resulting datasets.
print(f"Training data shape: {X_train.shape}")
print(f"Test data shape: {X_test.shape}")
print(f"Training targets shape: {y_train.shape}")
print(f"Test targets shape: {y_test.shape}")
```

```
Training data shape: (323, 13)
Test data shape: (81, 13)
Training targets shape: (323,)
Test targets shape: (81,)
```

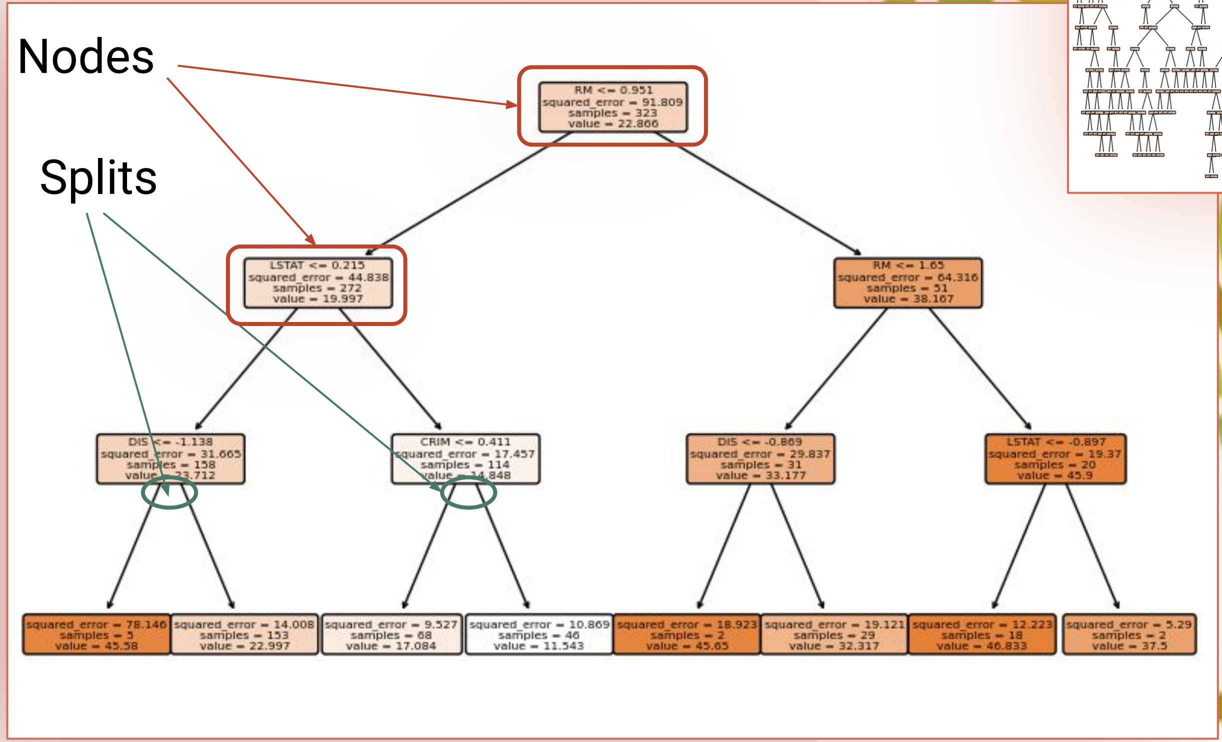
Epoch Loss

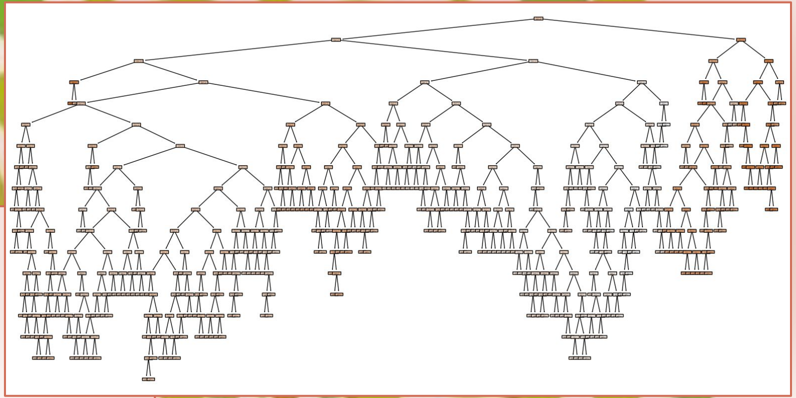
Loss is a number that tells us how much error or mistake a machine learning model made while learning from training data.

Validation loss measures how well a machine learning model performs on new data that it hasn't encountered during training. It helps assess the model's ability to make accurate predictions on unfamiliar examples.



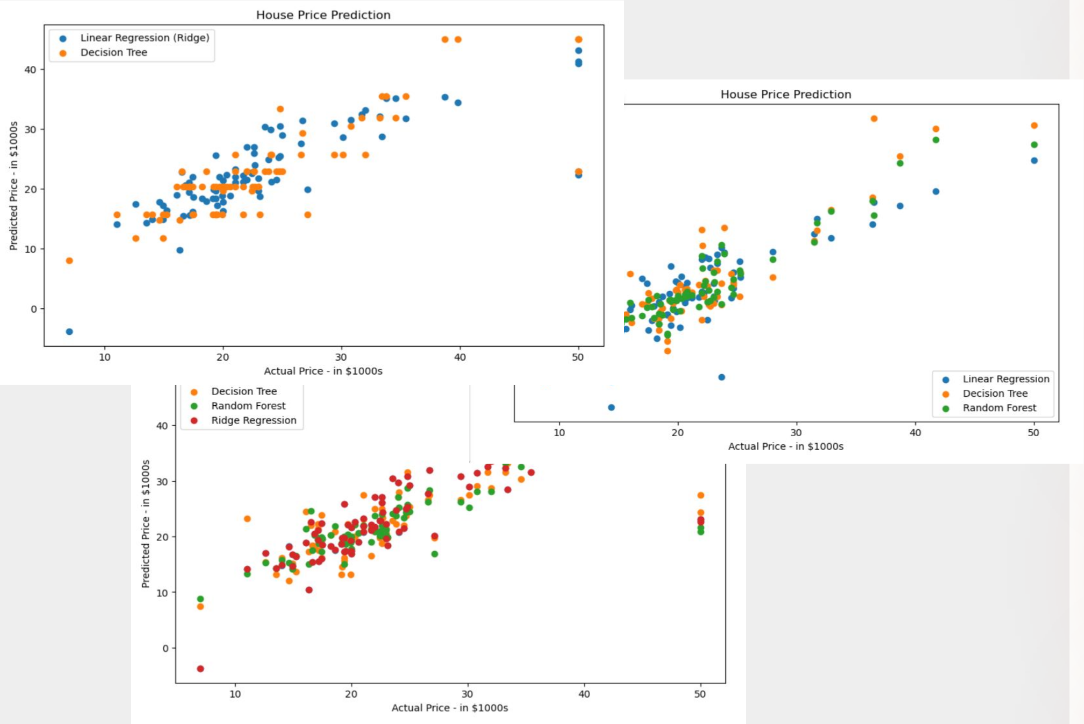
Decision Tree





Visualizing the decision tree model reveals how the model makes predictions based on features

Modeling



Comparing predictions from different models, such as linear regression, decision tree, and random forest, offers insights into their performance.

```
Linear Regression R-squared: 0.6205189361087049
Decision Tree R-squared: 0.1586075233676527l
Random Forest R-squared: 0.7149789927763945
```

```
Linear Regression R-squared: 0.6270849941673176
Decision Tree R-squared: 0.6906999395371678
Random Forest R-squared: 0.6686487964917727
```

```
Linear Regression with Ridge Regression
Mean Squared Error: 31.771340256023535
R-squared: 0.6233232154064134
```

```
Decision Tree Regression with Grid Search
Mean Squared Error: 29.741708956025036
R-squared: 0.6473862541650315
```

```
Linear Regression MSE: 23.195599256423012 R-squared: 0.7213535934621549
Decision Tree MSE: 17.735490196078434 R-squared: 0.7869453357642404
Random Forest MSE: 14.021053627450984 R-squared: 0.831566489575311
```

Presentation Summary

Insights

We delve into the decision tree model's mechanics, highlighting the process of optimization.

The depth of the tree dictates the complexity of decision-making, with deeper trees accommodating more intricate feature considerations.

Hyperparameters play a pivotal role in fine-tuning model performance, with techniques like grid search enabling systematic exploration of various parameter combinations.

Ridge regression serves as a crucial tool in combating overfitting, ensuring a more balanced and accurate model fit.

Conclusion

By balancing model complexity and predictive power, we pave the way for more accurate and reliable predictions, benefiting stakeholders in the real estate industry, policymakers, and prospective homebuyers, enabling informed decision-making and enhancing overall market efficiency.

QUESTIONS?

Resources

Purpose	Source
Dataset(s)	Boston housing dataset ‘from tensorflow.keras.datasets import boston_housing’
Dataset Description & Overview	The Boston Housing Dataset
Troubleshooting pyspark	Apache Spark CSV Files How to Read CSV File into PySpark DataFrame PySpark – Read CSV file into DataFrame
Project Examples	I built my own housing dataset 10 Real Estate Data Science Projects House Price Prediction
How it works	Full Resource List