# Statistical Machine Learning

Instructor : Benyamin Ghojogh

Fall 2023

Team 6 : Puja Saha, Om Bhosale, Sanchi Sanchi,
Parya Abadeh

# Project Title :Predictive Modeling and Analysis of Housing Prices

**Project Description:**

**The "Predictive Modeling and Analysis of Housing Prices" project is a comprehensive exploration of machine learning techniques and their application to predict housing prices. This project aims to provide valuable insights into the real estate market and help prospective homebuyers and sellers make informed decisions.**

# First Phase : Preprocessing

- Convert Categorical Data to numerical by one-hot encoding
- Selecting important features by feature selection algorithm named mutual_info_regression
    - Quantifies Relationship: Provides a numerical measure of the dependency between target and features.
    - Continuous Variables: Suited for scenarios where both target and features are continuous.
    - Information Gain: Higher mutual information implies more information gain, indicating a potentially stronger relationship.

# First Phase : Preprocessing

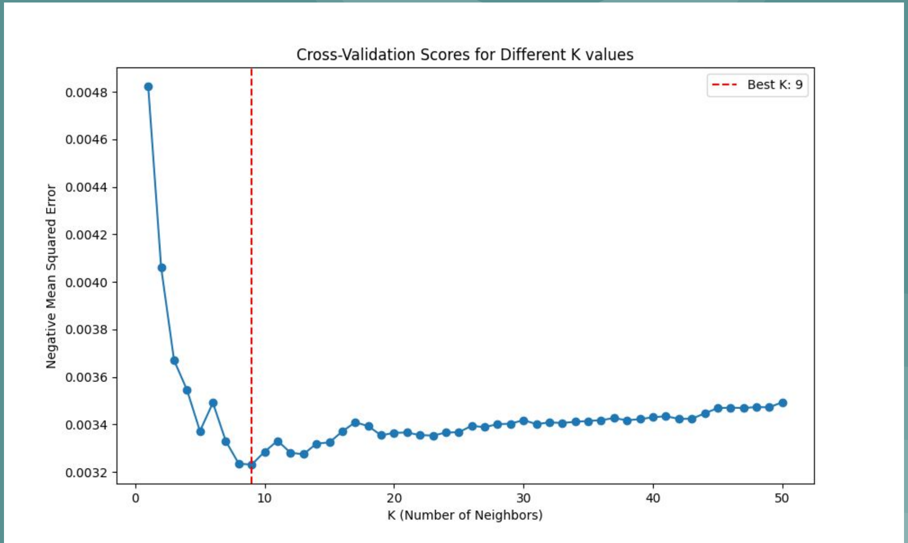
- Sample of Selected Features :
  - Feature,MI_Score
  - MSSubClass,0.20541657490230847
  - LotFrontage,0.16176817559279
  - OverallQual,0.2665880583532343
  - OverallCond,0.10550588082335333
  - YearBuilt,0.16905579849387298
  - YearRemodAdd,0.13009343474824586

# Second Phase : Split Dataset

- Splitting the new numerical data by only considering selected features into three part ( test : 10%, validation : 10% and train 80 %)

# Third Phase : Hyperparameter tuning for KNN

- Running KNeighborsRegressor algorithm for range k =1 to k =51 by considering scoring='neg_mean_squared_error' and 5 for cross-validation and find k =9 as the best



# Third Phase : Hyperparameter tuning for KNN

- Running KNN for k = 9 :
  - Mean Squared Error: 0.0033771100461378388
  - R-squared: 0.3215521935969837

# Algorithm Comparisons

# First Phase - Fetching, Splitting and Scaling

- Fetch the preprocessed data and split the data into training and testing sets.
- After splitting the data,standardize the features.The code  standardizes the features using StandardScaler, which can be important for regularized regression methods like Ridge and Lasso.
  - Standardization involves transforming the features such that they have a mean of 0 and a standard deviation of 1.
  - When using cross-validation, it's crucial to ensure that standardization is done within each fold to avoid data leakage. Use StandardScaler within a cross-validation pipeline

# Second Phase - Building the models

- In this phase, we select a range of regression algorithms, including Linear Regression, Ridge Regression, Lasso Regression to train the model.

- Also, evaluate each algorithm on the housing price dataset, using metrics such as Mean Squared Error (MSE) and R-squared (R2) to assess their predictive performance.

# Outputs

Linear Regression R-squared: -5.485663761154695e+26
Ridge Regression R-squared: -0.17292787917512165
Lasso Regression R-squared: -0.004796232081468288

Linear Regression MSE: 2.7855595849175894e+24
Ridge Regression MSE: 0.005955998468935661
Lasso Regression MSE: 0.005102244499532466

Best Ridge Hyperparameters: {'alpha': 10}
Best Lasso Hyperparameters: {'alpha': 0.1}

# Using selected features and running Random Forest Regression by Hyper-parameter tuning and using cross-validation for accuracy checking

**Hyperparameter Tuning:**

- Employed RandomizedSearchCV for hyperparameter tuning
- Explored a defined grid of hyperparameters
- Tuned parameters include:
    1. Number of estimators
    2. Maximum features
    3. Maximum depth
    4. Minimum samples split
    5. Minimum samples leaf
    6. Bootstrap

**Parallel Computation:**

- Parallelized computation using all available CPU cores (n_jobs=-1).

**Cross Validation:**

- Applied 5-fold cross-validation during hyperparameter tuning.

# Using selected features and running Random Forest Regression by Hyper-parameter tuning and using cross-validation for accuracy checking

Fitting 5 folds for each of 100 candidates, totalling 500 fits:

## Random Forest Regression Results on Validation Set:

Best Hyperparameters: {'n_estimators': 600, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 70, 'bootstrap': False}

Mean Squared Error: 0.0027919236
R-squared (R2): 0.4604896

## Random Forest Regression Results on Test Set:

Mean Squared Error: 0.00333537
R-squared (R2): 0.329937

# Using selected features and running Tree Regression by Hyper-parameter tuning and using cross-validation for accuracy checking

**Hyperparameter Tuning:**

- Employed RandomizedSearchCV for hyperparameter tuning.
- Explored a defined grid of hyperparameters:
    a. Maximum Depth
    b. Minimum Samples Leaf
    c. Minimum Samples Split

**Cross Validation:**

- Applied 5-fold cross-validation during hyperparameter tuning.

Om Bhosale

# Using selected features and running Tree Regression by Hyper-parameter tuning and using cross-validation for accuracy checking

Fitting 5 folds for each of 100 candidates, totalling 500 fits:

## Decision Tree Regression Results on Validation Set:

Best Hyperparameters: {'min_samples_split': 20, 'min_samples_leaf': 8, 'max_depth': 5}

Mean Squared Error: 0.00307601619
R-squared (R2): 0.40559167

## Decision Tree Regression Results on Test Set:

Mean Squared Error: 0.0042631988
R-squared (R2): 0.14354052

# Gradient Boosting and Support Vector Regression

Phase 1: Data Preparation & Feature Scaling

Phase 2: Model Initialization & Defining Parameters

Phase 3: Tuning Hyperparameters & Selecting Best

Phase 4: Model Training with the Best

Phase 5: Performance Analysis

# Phase 1

- **Data Preparation**
  - At first, processed numerical dataset was fetched. Splitting was done at this ratio (Train : Test = 8 : 2). Later 5-fold cross validation will be done on the training set.
- **Feature Scaling**
  - Scale both training and testing data using the fit-transform process to ensure that the features are standardized.

# Phase 2

- Model Initialization
  - Defining model either Support Vector or Gradient Boosting Regressor

- Defining Parameters
  - Defining a dictionary containing the hyperparameters to tune. Such as, for
    - SVR (Kernel , C, and Gamma)
    - GBR (number or estimator, learning rate, maximum depth)

# Phase 3

- **Tuning Hyperparameters**
  - Performed GridSearchCV to tune hyperparameter by searching through the defined parameter grid. It uses 5-fold cross-validation and negative mean squared error as the evaluation metric.

- **Selecting the Best for Training**
  - Fitted the GridSearchCV on the training data to find the best combination of hyperparameters.

# Phase 4

- **Model Training:** Fitted  the best estimator (model with optimized hyperparameters) on the entire training set.
- Best Parameters for Support Vector Regressor Model,

  Kernel = Linear and C= 0.1

- Best Parameters for Gradient Boosting Regressor Model,

  Number of Estimator = 300, Maximum Depth = 4 and Learning Rate = 0.05

# Phase 5

- **Performance Analysis**
  - Used the trained model to predict target values on the test set and calculated various evaluation metrics (RMSE, R2 Score) to assess the model's performance on the test set.
  - For SVR,

    RMSE = 0.07016370861293174 and R2 Score = 0.03051341456793988
  - For GBR,

    RMSE = 0.041047598203413506 and R2 Score = 0.6681878895913893