Bonus Experiments

These exercise ideas are exploratory and don't follow a specific order. Delve into the ones that pique your interest.

Error Distributions

- Investigate the distribution of errors in one or more models.
  - E.g., use the `cross_val_predict` function from `sklearn` and visualize a boxplot that shows the absolute differences between the predicted and actual values from validation sets.
- Display a scatter plot to compare predicted values against actual ones. Are there specific value ranges where your model consistently underperforms?

Metrics

- Familiarize yourself with various evaluation metrics and assess how your top-performing models measure against them:
  - Median Absolute Error (MAE).
  - Root Mean Squared Error (RMSE).
  - R-squared (R2).

Different K for Cross-validation

- Examine the impact of using various 'K' values in cross-validation. Consider that the optimal 'K' value isn't solely determined by the best performance!
  - Visualize performance metrics of a chosen model with different 'K' values for cross-validation splits. Depict both training and validation scores.
  - Chart the computation time associated with different 'K' values for cross-validation.

Learning Curves

- Conduct experiments using one or more of your top models, training them with different portions of data.
  - For instance, partition your training set into 10% training and 90% validation sets (cross-validation isn't required here). Do you get a good performance?
  - Analyze the outcomes and visualize the results for varying data portions like 10-90, 20-80, and so on.

Dimensionality Reduction

- Learn about Principal Component Analysis (PCA). There is no need to understand all the details.
  - Plot a chart depicting model performance by altering the number of principal components.
  - Draw conclusions from your findings.
- If RF has been explained in class: elect the most important features (e.g., based on RF feature importances or Lasso coefficients) and retrain your top models. Observe any changes in performance.
- Understand Feature Recursive Elimination (RFE) and apply it. Note any performance improvements or declines.
- Learn about and explore t-SNE. If computation time becomes an issue, reduce your dataset's size. Gauge if this changes model performance.
- Likewise, experiment with UMAP.

Linear Models

- Delve deeper into Linear Regression by exploring ISL section 3.1 (pages 70 to 80): https://drive.google.com/file/d/1ajFkHO6zjrdGNqhqW1jKBZdiNGh_8YQ1/view
- Learn about regularized linear models: Ridge, Lasso, and ElasticNet. Train them on your dataset and contrast their performance against other models.
  - "Introduction to Statistical Learning" and "Hands-on ML sklearn, keras & TF" books have good explanations about these models.
  - What are the differences in performance and training time with and without feature scaling?

Outliers

- Remove outliers from your target variable. Do your model's performance improve?
- Learn about the effect of transforming your target variable before training, and reverting it when predicting. For instance, a logarithmic transformation for training and using the exponential function for predictions. Consider using the `TransformedTargetRegressor` in `sklearn`.

Tree-based Models

(Only proceed if trees have been covered in class)

- Understand the concept of the Out-of-bag (OOB) estimate. Implement it in your models and compare the results with validation scores.
- Learn about Extratrees and apply it.

Bagging

(Only proceed if bagging has been discussed in class)

- Use Bagging with ML algorithms other than Decision Trees, such as LR or kNN. Check `BaggingRegressor` in `sklearn`. Interpret the outcomes and theorize why certain results occurred. Consider ways to enhance performance.

Boosting

(Only proceed if boosting has been explained in class)

- Employ Gradient Boosting with algorithms apart from Decision Trees, like LR or kNN. Check `GradientBoostingRegressor` in `sklearn`. Interpret the outcomes and theorize why certain results occurred. Consider ways to enhance performance.
- Learn about AdaBoost and apply it.

Other Ensembling Techniques

- Learn about stacking and apply it.
- Learn about blending and apply it.

Interpretability

- Learn about Permutation Feature Importance and apply it to understand the contribution of each feature to the prediction. Check `permutation_importance` in `sklearn`.
- Likewise, learn and experiment with SHAP (SHapley Additive exPlanations).
- Likewise, learn and experiment with LIME .

Hyperparameter Optimization

(Only proceed if hyperparameter optimization has been introduced in class)

- Learn about Bayesian optimization and apply it.

Automatic ML

- Explore automated machine learning using libraries such as Auto-sklearn, H2O AutoML, Optuna, TPOT, among others.

Full Pipeline

- Envision an automated system for regression that periodically receives new data. Design a comprehensive pipeline to emulate this process: data preprocessing, train/test splitting, hyperparameter tuning, validation, testing, and monitoring (including performance evaluation logs and potential failure alerts).
- Model Drift Detection: Implement mechanisms to detect if the model's performance degrades over time as new data becomes available.