Bonus Experiments

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

Metrics

- Familiarize yourself with various classification metrics and assess how your top-performing models measure against them:
  - Accuracy
  - Precision
  - Recall
  - Balanced Accuracy
  - F1 score

Decision threshold

- The decision threshold specifies at which probability the prediction is a 0 or is a 1.
- By default, most models set it to 0.5.
- E.g., Samples with a probability of 0.45 (45%) of being a 1 are set to 0, and features with probability of 0.55 (55%) of being a 1 are set to 1.
- Plot a chart where the x-axis is the decision threshold (between 0 and 1) and the y-axis the model's performance (e.g., plot precision and recall). You are going to see how the predictions vary when changing this threshold. You have to implement this process manually by using the `predict_proba()` function and calculating the metrics.

ROC-AUC

- Search information and understand what is the ROC-AUC curve.
- Plot it for your models.

Precision-Recall Curve

- Search information and understand what is the ROC-AUC curve.
- Plot it for your models.

Different K for Cross-validation

- Examine the impact of using various 'K' values in cross-validation.
  - Visualize performance metrics of a chosen model with different 'K' values.
  - 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?

Tree-based Models

- Understand the concept of the Out-of-bag (OOB) estimate. Check in in your RF model and compare the results with validation scores.
- Learn about Extratrees and apply it.
- Try other libraries better suited for tree-based models than `sklearn`. E.g., `XGBoost`, `LightGBM`, or `CatBoost`.

Bagging

- 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

- 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.
- Learn about cascading 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

- Learn about Bayesian optimization and apply it.

More Features

- Derive (from the existing attributes) more features that can be useful for prediction.
- Why is it useful to transform some features by computing their logarithm? Research, think and try it.
- Why is it useful to transform some features by computing a polynomial? Research, think and try 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.