Homework 1

2 points

1. Move code to a library

Jupyter Notebooks are not good for managing code. They are best for visualization and quick iteration. So, we'll move useful code to a library. Later, we can import the library and call its methods from the notebooks.

Useful things to move to the library:

1. Data preprocessing: load from csv, clean up data.
2. Dataset split.
3. Metrics and score calculation.

Steps:

1. Create a Python package `csc665`. It's just a subdirectory, with an empty file `__init__.py` in it.
2. Create `features.py` in the `csc665` subdirectory and implement the following functions:
   A. `def train_test_split(X, y, test_size, shuffle, random_state=None):`
      - *X*, *y* - features and the target variable.
      - test_size - between 0 and 1 - how much to allocate to the test set; the rest goes to the train set.
      - shuffle - if True, shuffle the dataset, otherwise not.
      - random_state, integer; if None, then results are random, otherwise fixed to a given seed.
      - Example:
        - X_train, X_test, y_train, y_test = train_test_split(feat_df, y, 0.3, True, 12)
   B. `create_categories(df, list_columns)`
      - Converts values, **in-place**, in the columns passed in the *list_columns* to numerical values. Follow the same approach: "string" -> category -> code.
      - Replace values in df, **in-place**.
   C. `X, y = preprocess_ver_1(csv_df)`
      - Apply the feature transformation steps to the dataframe, return new `X` and `y` for entire dataset. Do not modify the original `csv_df`.
        - Remove all rows with NA values
        - Convert datetime to a number
        - Convert all strings to numbers.
        - Split the dataframe into X and y and return these.
3. Create `metrics.py`:
   A. `def mse(y_predicted, y_true)` - return Mean-Squared Error.
   B. `def rmse(y_predicted, y_true)` - return Root Mean-Squared Error.
   C. `def rsq(y_predicted, y_true)` - return R^2.

1.5 Update the In-class Notebook

Copy the in-class notebook, and replace all relevant data processing and feature calculations with your own functions.

2. Evaluate Impact of the Number of Trees

Evaluate the R^2 score as a function of the number of trees (i.e. n_estimators) used in the random forest. That is, set the number of trees to 1, 5, 10, 20, ..., 200, and measure R^2 on *both* the train and test sets.

Plot both train and test R^2 scores as the function on the number of trees used in the random forest model.

Analyze the result. Is it overfitting / underfitting? How overfitting / underfitting changes with the number of trees?

3. Results

Commit the folder with you library and the notebook to your **private Classroom Github** repository.

The final version must be submmited for review **before Thursday, Feb 14, 12PM**.