{"cells":[{"metadata":{},"cell_type":"markdown","source":"**This notebook is an exercise in the [Intermediate Machine Learning]
(https://www.kaggle.com/learn/intermediate-machine-learning) course. You can reference the tutorial at [this link]
(https://www.kaggle.com/alexisbcook/cross-validation).**\n\n---\n"},{"metadata":{},"cell_type":"markdown","source":"In this
exercise, you will leverage what you've learned to tune a machine learning model with **cross-validation**.\n\n# Setup\n\nThe
questions below will give you feedback on your work. Run the following cell to set up the feedback system."},{"metadata":
{"trusted":false},"cell_type":"code","source":"# Set up code checking\nimport os\nif not os.path.exists(\"../input/train.csv\"):\n
os.symlink(\"../input/home-data-for-ml-course/train.csv\", \"../input/train.csv\") \n   os.symlink(\"../input/home-data-for-ml-
course/test.csv\", \"../input/test.csv\") \nfrom learntools.core import binder\nbinder.bind(globals())\nfrom
learntools.ml_intermediate.ex5 import *\nprint(\"Setup Complete\")","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"You will work with the [Housing Prices Competition for Kaggle Learn Users]
(https://www.kaggle.com/c/home-data-for-ml-course) from the previous exercise. \n\n![Ames Housing dataset image]
(https://i.imgur.com/lTJVG4e.png)\n\nRun the next code cell without changes to load the training and validation sets in `X_train`,
`X_valid`, `y_train`, and `y_valid`. The test set is loaded in `X_test`.\n\nFor simplicity, we drop categorical variables."},
{"metadata":{"trusted":false},"cell_type":"code","source":"import pandas as pd\nfrom sklearn.model_selection import
train_test_split\n\n# Read the data\ntrain_data = pd.read_csv('../input/train.csv', index_col='Id')\ntest_data =
pd.read_csv('../input/test.csv', index_col='Id')\n\n# Remove rows with missing target, separate target from
predictors\ntrain_data.dropna(axis=0, subset=['SalePrice'], inplace=True)\ny = train_data.SalePrice
\ntrain_data.drop(['SalePrice'], axis=1, inplace=True)\n\n# Select numeric columns only\nnumeric_cols = [cname for cname in
train_data.columns if train_data[cname].dtype in ['int64', 'float64']]\nX = train_data[numeric_cols].copy()\nX_test =
test_data[numeric_cols].copy()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"Use the next
code cell to print the first several rows of the data."},{"metadata":
{"trusted":false},"cell_type":"code","source":"X.head()","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"So far, you've learned how to build pipelines with scikit-learn. For instance, the pipeline
below will use [`SimpleImputer()`](https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html) to replace
missing values in the data, before using [`RandomForestRegressor()`](https://scikit-
learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html) to train a random forest model to make predictions.
We set the number of trees in the random forest model with the `n_estimators` parameter, and setting `random_state` ensures
reproducibility."},{"metadata":{"trusted":false},"cell_type":"code","source":"from sklearn.ensemble import
RandomForestRegressor\nfrom sklearn.pipeline import Pipeline\nfrom sklearn.impute import SimpleImputer\n\nmy_pipeline =
Pipeline(steps=[\n   ('preprocessor', SimpleImputer()),\n   ('model', RandomForestRegressor(n_estimators=50,
random_state=0))\n])","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"You have also learned
how to use pipelines in cross-validation. The code below uses the [`cross_val_score()`](https://scikit-
learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html) function to obtain the mean absolute error (MAE),
averaged across five different folds. Recall we set the number of folds with the `cv` parameter."},{"metadata":
{"trusted":false},"cell_type":"code","source":"from sklearn.model_selection import cross_val_score\n\n# Multiply by -1 since
sklearn calculates *negative* MAE\nscores = -1 * cross_val_score(my_pipeline, X, y,\n   cv=5,\n
scoring='neg_mean_absolute_error')\n\nprint(\"Average MAE score:\", scores.mean())","execution_count":null,"outputs":[]},
{"metadata":{},"cell_type":"markdown","source":"# Step 1: Write a useful function\n\nIn this exercise, you'll use cross-validation
to select parameters for a machine learning model.\n\nBegin by writing a function `get_score()` that reports the average (over
three cross-validation folds) MAE of a machine learning pipeline that uses:\n- the data in `X` and `y` to create folds,\n-
`SimpleImputer()` (with all parameters left as default) to replace missing values, and\n- `RandomForestRegressor()` (with
`random_state=0`) to fit a random forest model.\n\nThe `n_estimators` parameter supplied to `get_score()` is used when setting the
number of trees in the random forest model. "},{"metadata":{"trusted":false},"cell_type":"code","source":"def
get_score(n_estimators):\n   \"\"\"Return the average MAE over 3 CV folds of random forest model.\n   \n   Keyword argument:\n

n_estimators -- the number of trees in the forest\n   \"\"\"\n   my_pipeline = Pipeline(steps=[\n   ('preprocessor', SimpleImputer()),\n   ('model', RandomForestRegressor(n_estimators, random_state=0))\n   ])\n   \n   scores = -1 * cross_val_score(my_pipeline, X, y,\n   cv=3,\n scoring='neg_mean_absolute_error')\n   return scores.mean()\n\n# Check your answer\nstep_1.check()","execution_count":null,"outputs":[]},{"metadata":{"trusted":false},"cell_type":"code","source":"# Lines below will give you a hint or solution code\n#step_1.hint()\n#step_1.solution()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"# Step 2: Test different parameter values\n\nNow, you will use the function that you defined in Step 1 to evaluate the model performance corresponding to eight different values for the number of trees in the random forest: 50, 100, 150, ..., 300, 350, 400.\n\nStore your results in a Python dictionary `results`, where `results[i]` is the average MAE returned by `get_score(i)`."},{"metadata":{"trusted":false},"cell_type":"code","source":"results = {}\nfor i in range(1,9):\n results[50*i] = get_score(50*i)\n\n\n\nstep_2.check()","execution_count":null,"outputs":[]},{"metadata":{"trusted":false},"cell_type":"code","source":"# Lines below will give you a hint or solution code\n#step_2.hint()\n#step_2.solution()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"Use the next cell to visualize your results from Step 2. Run the code without changes."},{"metadata":{"trusted":false},"cell_type":"code","source":"import matplotlib.pyplot as plt\n%matplotlib inline\n\nplt.plot(list(results.keys()), list(results.values()))\nplt.show()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"# Step 3: Find the best parameter value\n\nGiven the results, which value for `n_estimators` seems best for the random forest model? Use your answer to set the value of `n_estimators_best`."},{"metadata":{"trusted":false},"cell_type":"code","source":"n_estimators_best = min(results, key=results.get)\n\n# Check your answer\nstep_3.check()","execution_count":null,"outputs":[]},{"metadata":{"trusted":false},"cell_type":"code","source":"# Lines below will give you a hint or solution code\n#step_3.hint()\n#step_3.solution()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"In this exercise, you have explored one method for choosing appropriate parameters in a machine learning model. \n\nIf you'd like to learn more about [hyperparameter optimization](https://en.wikipedia.org/wiki/Hyperparameter_optimization), you're encouraged to start with **grid search**, which is a straightforward method for determining the best _combination_ of parameters for a machine learning model. Thankfully, scikit-learn also contains a built-in function [`GridSearchCV()`](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) that can make your grid search code very efficient!\n\n# Keep going\n\nContinue to learn about **[gradient boosting](https://www.kaggle.com/alexisbcook/xgboost)**, a powerful technique that achieves state-of-the-art results on a variety of datasets."},{"metadata":{},"cell_type":"markdown","source":"---\n\n\n\n\n*Have questions or comments? Visit the [Learn Discussion forum](https://www.kaggle.com/learn-forum/161289) to chat with other Learners.*"}],"metadata":{"kernelspec":{"language":"python","display_name":"Python 3","name":"python3"},"language_info":{"pygments_lexer":"ipython3","nbconvert_exporter":"python","version":"3.6.4","file_extension":".py","codemirror_mode":{"name":"ipython","version":3},"name":"python","mimetype":"text/x-python"}},"nbformat":4,"nbformat_minor":4}