```
{"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/xgboost).**\n\n---\n"},{"metadata":{},"cell_type":"markdown","source":"In this exercise, you
will use your new knowledge to train a model with **gradient boosting**.\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.ex6 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) dataset 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`."},{"metadata":
{"trusted":false},"cell_type":"code","source":"import pandas as pd\nfrom sklearn.model_selection import train_test_split\n\n# Read
the data\nX = pd.read_csv('../input/train.csv', index_col='Id')\nX_test_full = pd.read_csv('../input/test.csv',
index_col='Id')\n\n# Remove rows with missing target, separate target from predictors\nX.dropna(axis=0, subset=['SalePrice'],
inplace=True)\ny = X.SalePrice              \nX.drop(['SalePrice'], axis=1, inplace=True)\n\n# Break off validation set from
training data\nX_train_full, X_valid_full, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2,\n
random_state=0)\n\n# \"Cardinality\" means the number of unique values in a column\n# Select categorical columns with relatively
low cardinality (convenient but arbitrary)\nlow_cardinality_cols = [cname for cname in X_train_full.columns if
X_train_full[cname].nunique() < 10 and \n                        X_train_full[cname].dtype == \"object\"]\n\n# Select numeric
columns\nnumeric_cols = [cname for cname in X_train_full.columns if X_train_full[cname].dtype in ['int64', 'float64']]\n\n# Keep
selected columns only\nmy_cols = low_cardinality_cols + numeric_cols\nX_train = X_train_full[my_cols].copy()\nX_valid =
X_valid_full[my_cols].copy()\nX_test = X_test_full[my_cols].copy()\n\n# One-hot encode the data (to shorten the code, we use
pandas)\nX_train = pd.get_dummies(X_train)\nX_valid = pd.get_dummies(X_valid)\nX_test = pd.get_dummies(X_test)\nX_train, X_valid =
X_train.align(X_valid, join='left', axis=1)\nX_train, X_test = X_train.align(X_test, join='left',
axis=1)","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"# Step 1: Build model\n\n### Part
A\n\nIn this step, you'll build and train your first model with gradient boosting.\n\n- Begin by setting `my_model_1` to an XGBoost
model.  Use the [XGBRegressor](https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBRegressor) class, and set
the random seed to 0 (`random_state=0`).  **Leave all other parameters as default.**\n- Then, fit the model to the training data in
`X_train` and `y_train`."},{"metadata":{"trusted":false},"cell_type":"code","source":"from xgboost import XGBRegressor\n\n# Define
the model\nmy_model_1 = XGBRegressor(random_state=0)\n\n# Fit the model\nmy_model_1.fit(X_train, y_train)\n\n# Check your
answer\nstep_1.a.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.a.hint()\n#step_1.a.solution()","execution_count":null,"outputs":[]},
{"metadata":{},"cell_type":"markdown","source":"### Part B\n\nSet `predictions_1` to the model's predictions for the validation
data.  Recall that the validation features are stored in `X_valid`."},{"metadata":
{"trusted":false},"cell_type":"code","source":"from sklearn.metrics import mean_absolute_error\n\n# Get predictions\npredictions_1
= my_model_1.predict(X_valid)\n\n# Check your answer\nstep_1.b.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.b.hint()\n#step_1.b.solution()","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"### Part C\n\nFinally, use the `mean_absolute_error()` function to calculate the mean absolute
error (MAE) corresponding to the predictions for the validation set.  Recall that the labels for the validation data are stored in
`y_valid`."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Calculate MAE\nmae_1 = mean_absolute_error(predictions_1,
y_valid)\n\n# Uncomment to print MAE\nprint(\"Mean Absolute Error:\" , mae_1)\n\n# Check your
answer\nstep_1.c.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.c.hint()\n#step_1.c.solution()","execution_count":null,"outputs":[]},
{"metadata":{},"cell_type":"markdown","source":"# Step 2: Improve the model\n\nNow that you've trained a default model as baseline,
it's time to tinker with the parameters, to see if you can get better performance!\n- Begin by setting `my_model_2` to an XGBoost
model, using the [XGBRegressor](https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBRegressor) class. Use
what you learned in the previous tutorial to figure out how to change the default parameters (like `n_estimators` and
`learning_rate`) to get better results.\n- Then, fit the model to the training data in `X_train` and `y_train`.\n- Set
`predictions_2` to the model's predictions for the validation data. Recall that the validation features are stored in
`X_valid`.\n- Finally, use the `mean_absolute_error()` function to calculate the mean absolute error (MAE) corresponding to the
predictions on the validation set. Recall that the labels for the validation data are stored in `y_valid`.\n\nIn order for this
step to be marked correct, your model in `my_model_2` must attain lower MAE than the model in `my_model_1`. "},{"metadata":
{"trusted":false},"cell_type":"code","source":"# Define the model\nmy_model_2 = XGBRegressor(n_estimators=1000,
learning_rate=0.05)\n\n# Fit the model\nmy_model_2.fit(X_train, y_train)\n\n# Get predictions\npredictions_2 =
my_model_2.predict(X_valid)\n\n# Calculate MAE\nmae_2 = mean_absolute_error(predictions_2, y_valid)\nprint(\"Mean Absolute Error:\"
, mae_2)\n\n# Check your answer\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":"#
Step 3: Break the model\n\nIn this step, you will create a model that performs worse than the original model in Step 1. This will
help you to develop your intuition for how to set parameters. You might even find that you accidentally get better performance,
which is ultimately a nice problem to have and a valuable learning experience!\n- Begin by setting `my_model_3` to an XGBoost
model, using the [XGBRegressor](https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBRegressor) class. Use
what you learned in the previous tutorial to figure out how to change the default parameters (like `n_estimators` and
`learning_rate`) to design a model to get high MAE.\n- Then, fit the model to the training data in `X_train` and `y_train`.\n- Set
`predictions_3` to the model's predictions for the validation data. Recall that the validation features are stored in
`X_valid`.\n- Finally, use the `mean_absolute_error()` function to calculate the mean absolute error (MAE) corresponding to the
predictions on the validation set. Recall that the labels for the validation data are stored in `y_valid`.\n\nIn order for this
step to be marked correct, your model in `my_model_3` must attain higher MAE than the model in `my_model_1`. "},{"metadata":
{"trusted":false},"cell_type":"code","source":"# Define the model\nmy_model_3 = XGBRegressor(n_estimators=1)\n\n# Fit the
model\nmy_model_3.fit(X_train, y_train)\n\n# Get predictions\npredictions_3 = my_model_3.predict(X_valid)\n\n# Calculate MAE\nmae_3
= mean_absolute_error(predictions_3, y_valid)\nprint(\"Mean Absolute Error:\" , mae_3)\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":"# Keep going\n\nContinue to learn about **[data leakage]
(https://www.kaggle.com/alexisbcook/data-leakage)**. This is an important issue for a data scientist to understand, and it has the
potential to ruin your models in subtle and dangerous ways!"},{"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}