{"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/missing-values).**\n\n---\n"},{"metadata":{},"cell_type":"markdown","source":"Now it's your turn to test your new knowledge of **missing values** handling. You'll probably find it makes a big difference.\n\n# Setup\n\nThe questions 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.ex2 import *\nprint(\"Setup Complete\")","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"In this exercise, you will work with data from the [Housing Prices Competition for Kaggle Learn Users](https://www.kaggle.com/c/home-data-for-ml-course). \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_full = 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_full.dropna(axis=0, subset=['SalePrice'], inplace=True)\ny = X_full.SalePrice\nX_full.drop(['SalePrice'], axis=1, inplace=True)\n\n# To keep things simple, we'll use only numerical predictors\nX = X_full.select_dtypes(exclude=['object'])\nX_test = X_test_full.select_dtypes(exclude=['object'])\n\n# Break off validation set from training data\nX_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2,\n   random_state=0)","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"Use the next code cell to print the first five rows of the data."},{"metadata":{"trusted":false},"cell_type":"code","source":"X_train.head()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"You can already see a few missing values in the first several rows. In the next step, you'll obtain a more comprehensive understanding of the missing values in the dataset.\n\n# Step 1: Preliminary investigation\n\nRun the code cell below without changes."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Shape of training data (num_rows, num_columns)\nprint(X_train.shape)\n\n# Number of missing values in each column of training data\nmissing_val_count_by_column = (X_train.isnull().sum())\nprint(missing_val_count_by_column[missing_val_count_by_column > 0])","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"### Part A\n\nUse the above output to answer the questions below."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Fill in the line below: How many rows are in the training data?\nnum_rows = 1168\n\n# Fill in the line below: How many columns in the training data\n# have missing values?\nnum_cols_with_missing = 3\n\n# Fill in the line below: How many missing entries are contained in \n# all of the training data?\ntot_missing = 276\n\n# Check your answers\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\nConsidering your answers above, what do you think is likely the best approach to dealing with the missing values?"},{"metadata":{"trusted":false},"cell_type":"code","source":"# Check your answer (Run this code cell to receive credit!)\nstep_1.b.check()","execution_count":null,"outputs":[]},{"metadata":{"trusted":false},"cell_type":"code","source":"#step_1.b.hint()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"To compare different approaches to dealing with missing values, you'll use the same `score_dataset()` function from the tutorial. This function reports the [mean absolute error](https://en.wikipedia.org/wiki/Mean_absolute_error) (MAE) from a random forest model."},{"metadata":{"trusted":false},"cell_type":"code","source":"from sklearn.ensemble import RandomForestRegressor\nfrom sklearn.metrics import mean_absolute_error\n\n# Function for comparing different approaches\ndef score_dataset(X_train, X_valid, y_train, y_valid):\n   model = RandomForestRegressor(n_estimators=100, random_state=0)\n   model.fit(X_train, y_train)\n   preds =

model.predict(X_valid)\n   return mean_absolute_error(y_valid, preds)","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"# Step 2: Drop columns with missing values\n\nIn this step, you'll preprocess the data in
`X_train` and `X_valid` to remove columns with missing values. Set the preprocessed DataFrames to `reduced_X_train` and
`reduced_X_valid`, respectively. "},{"metadata":{"trusted":false},"cell_type":"code","source":"# Fill in the line below: get names
of columns with missing values\ncol_with_missing_values=[col for col in X_train.columns\n   if
X_train[col].isnull().any()] # Your code here\n\n# Fill in the lines below: drop columns in training and validation
data\nreduced_X_train = X_train.drop(col_with_missing_values,axis=1)\nreduced_X_valid =
X_valid.drop(col_with_missing_values,axis=1)\n\n# Check your answers\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":"Run
the next code cell without changes to obtain the MAE for this approach."},{"metadata":
{"trusted":false},"cell_type":"code","source":"print(\"MAE (Drop columns with missing
values):\")\nprint(score_dataset(reduced_X_train, reduced_X_valid, y_train, y_valid))","execution_count":null,"outputs":[]},
{"metadata":{},"cell_type":"markdown","source":"# Step 3: Imputation\n\n### Part A\n\nUse the next code cell to impute missing
values with the mean value along each column. Set the preprocessed DataFrames to `imputed_X_train` and `imputed_X_valid`. Make
sure that the column names match those in `X_train` and `X_valid`."},{"metadata":
{"trusted":false},"cell_type":"code","source":"from sklearn.impute import SimpleImputer\n\n# Fill in the lines below:
imputation\nmy_imputer= SimpleImputer(strategy=\"mean\") # Your code here\nimputed_X_train =
pd.DataFrame(my_imputer.fit_transform(X_train))\nimputed_X_valid = pd.DataFrame(my_imputer.transform(X_valid))\n\n# Fill in the
lines below: imputation removed column names; put them back\nimputed_X_train.columns = X_train.columns\nimputed_X_valid.columns =
X_valid.columns\n\n# Check your answers\nstep_3.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_3.a.hint()\n#step_3.a.solution()","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"Run the next code cell without changes to obtain the MAE for this approach."},{"metadata":
{"trusted":false},"cell_type":"code","source":"print(\"MAE (Imputation):\")\nprint(score_dataset(imputed_X_train, imputed_X_valid,
y_train, y_valid))","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"### Part B\n\nCompare the
MAE from each approach. Does anything surprise you about the results? Why do you think one approach performed better than the
other?"},{"metadata":{"trusted":false},"cell_type":"code","source":"# Check your answer (Run this code cell to receive
credit!)\nstep_3.b.check()","execution_count":null,"outputs":[]},{"metadata":
{"trusted":false},"cell_type":"code","source":"#step_3.b.hint()","execution_count":null,"outputs":[]},{"metadata":
{},"cell_type":"markdown","source":"# Step 4: Generate test predictions\n\nIn this final step, you'll use any approach of your
choosing to deal with missing values. Once you've preprocessed the training and validation features, you'll train and evaluate a
random forest model. Then, you'll preprocess the test data before generating predictions that can be submitted to the
competition!\n\n### Part A\n\nUse the next code cell to preprocess the training and validation data. Set the preprocessed
DataFrames to `final_X_train` and `final_X_valid`. **You can use any approach of your choosing here!** in order for this step to
be marked as correct, you need only ensure:\n- the preprocessed DataFrames have the same number of columns,\n- the preprocessed
DataFrames have no missing values, \n- `final_X_train` and `y_train` have the same number of rows, and\n- `final_X_valid` and
`y_valid` have the same number of rows."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Preprocessed training and
validation features\nfinal_imputer = SimpleImputer(strategy='median')\nfinal_X_train =
pd.DataFrame(final_imputer.fit_transform(X_train))\nfinal_X_valid = pd.DataFrame(final_imputer.transform(X_valid))\n\n# Restoring
column names\nfinal_X_train.columns = X_train.columns\nfinal_X_valid.columns = X_valid.columns\n\n\n\n# Check your
answers\nstep_4.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_4.a.hint()\n#step_4.a.solution()","execution_count":null,"outputs":[]},
{"metadata":{},"cell_type":"markdown","source":"Run the next code cell to train and evaluate a random forest model. (*Note that we
don't use the `score_dataset()` function above, because we will soon use the trained model to generate test predictions!*)"},

```
{"metadata":{"trusted":false},"cell_type":"code","source":"# Define and fit model\nmodel = RandomForestRegressor(n_estimators=100, random_state=0)\nmodel.fit(final_X_train, y_train)\n\n# Get validation predictions and MAE\npreds_valid = model.predict(final_X_valid)\nprint(\"MAE (Your approach):\")\nprint(mean_absolute_error(y_valid, preds_valid))","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"### Part B\n\nUse the next code cell to preprocess your test data.  Make sure that you use a method that agrees with how you preprocessed the training and validation data, and set the preprocessed test features to `final_X_test`.\n\nThen, use the preprocessed test features and the trained model to generate test predictions in `preds_test`.\n\nIn order for this step to be marked correct, you need only ensure:\n- the preprocessed test DataFrame has no missing values, and\n- `final_X_test` has the same number of rows as `X_test`."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Fill in the line below: preprocess test data\nfinal_X_test = pd.DataFrame(my_imputer.transform(X_test))\n\n\n\n# Fill in the line below: get test predictions\npreds_test = model.predict(final_X_test)\n\nstep_4.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_4.b.hint()\n#step_4.b.solution()","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"Run the next code cell without changes to save your results to a CSV file that can be submitted directly to the competition."},{"metadata":{"trusted":false},"cell_type":"code","source":"# Save test predictions to file\noutput = pd.DataFrame({'Id': X_test.index,\n                       'SalePrice': preds_test})\noutput.to_csv('submission.csv', index=False)","execution_count":null,"outputs":[]},{"metadata":{},"cell_type":"markdown","source":"# Submit your results\n\nOnce you have successfully completed Step 4, you're ready to submit your results to the leaderboard!  (_You also learned how to do this in the previous exercise.  If you need a reminder of how to do this, please use the instructions below._)  \n\nFirst, you'll need to join the competition if you haven't already.  So open a new window by clicking on [this link](https://www.kaggle.com/c/home-data-for-ml-course).  Then click on the **Join Competition** button.\n\n![join competition image](https://i.imgur.com/wLmFtH3.png)\n\nNext, follow the instructions below:\n1. Begin by clicking on the blue **Save Version** button in the top right corner of the window.  This will generate a pop-up window.  \n2. Ensure that the **Save and Run All** option is selected, and then click on the blue **Save** button.\n3. This generates a window in the bottom left corner of the notebook.  After it has finished running, click on the number to the right of the **Save Version** button.  This pulls up a list of versions on the right of the screen.  Click on the ellipsis **(...)** to the right of the most recent version, and select **Open in Viewer**.  This brings you into view mode of the same page. You will need to scroll down to get back to these instructions.\n4. Click on the **Output** tab on the right of the screen.  Then, click on the file you would like to submit, and click on the blue **Submit** button to submit your results to the leaderboard.\n\nYou have now successfully submitted to the competition!\n\nIf you want to keep working to improve your performance, select the blue **Edit** button in the top right of the screen. Then you can change your code and repeat the process. There's a lot of room to improve, and you will climb up the leaderboard as you work.\n\n\n# Keep going\n\nMove on to learn what **[categorical variables](https://www.kaggle.com/alexisbcook/categorical-variables)** are, along with how to incorporate them into your machine learning models.  Categorical variables are very common in real-world data, but you'll get an error if you try to plug them into your models without processing them first!"},{"metadata":{},"cell_type":"markdown","source":"---\n\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}
```