lab6

October 7, 2019

1 House price prediction with Ames Housing Dataset

In this lab exercise, you will prepare your data in a way that it is suitable for a machine learning algorithm. You will achive this by first exploring the data and performing feature transformations on provided dataset of house price prediction ML problem. To do this exercise, you will need to have installed Python and required packages of external modules. Check installation guide document for more installation details:

- **scipy**: For advanced mathematical routines
- **numpy**: For N-dimensional array manipulation
- **pandas alias np**: For data analysis and data structures in DataFrames (i.e. tabular data with rows and columns)
- **matplotlib.pyplot alias plt**: For 2D plotting
- **sklearn**: For machine learning algorithms

You will write code and submit a python file called 'housePricePrediction_giveYourLastName.py' as you perform the following steps in this exercise:

- *Step 0: Import external modules*
- *Step 1: Load dataset*
- *Step 2: Visualize and explore dataset in order to gain insights*
- *Step 3: Transform and select features*
- *Step 4: Train a model*
- *Step 5: Make predictions and evaluate the model*

1.1 Files included in this exercise and assignment:

- **Ames_Housing_dataset**
  - train.csv
  - data_description.txt
- **Installation guide with example implementaion of lab1**
- **housePricePrediction_giveYourLastName.py**

1.2 Where to get help:

This exercise uses Python and supporting ML related packages. Installation guidelines have been provided and you can refer to the document for all installations. To learn more about ML with Python, here are some useful sources:

- **Working with Pandas**:
  - Load data: https://pandas.pydata.org/pandas-docs/stable/reference/io.html#flat-file
  - DataFrame: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
  - Explore data: https://pandas.pydata.org/pandas-docs/stable/reference/io.html#flat-file
- **Working with Pandas and Matplotlib**: https://pandas.pydata.org/pandas-docs/version/0.13/visualization.html
  - Visualize data e.g., scatterplot: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html
  - Scatter Matrix: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.scatter_matrix.html
  - Pivot table: https://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html
- **Working with sklearn**:
  - train_test_split: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
  - ML algorithms: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
  - ML evaluation metrics: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error

2 Load Dataset

The ML problem is that of predicting house prices based on the provided Ames housing dataset. Edit the provided Python file called 'housePricePrediction_giveYourLastName.py' with your preferred IDE. In this first task, you will load dataset `train.csv` to a Panda's DataFrame object named `train_house_data`. You will then confirm that the data is loaded by looking at data structure.

TASK 1. Write, in your Python file, code to load data using Pandas `read_csv()`, which will return a DataFrame that you can explore. To briefly look at data structure, check and print:

- Data size with `DataFrame.shape`
- Data summary with `DataFrame.info()`
- First five rows with `DataFrame.head()`
- Summary of numeric features values (mean, min, max, std etc) with `DataFrame.describe()`
- Values of a categorical feature (Street) with `DataFrame['column'].value_counts()`

```
Data shape after removing Id: (1460, 80)
```

There are 1460 instances in the train dataset. Notice that some features have missing values that need to be take care of if you are to use the feature in your model. For example, 'Alley' feature has only 91 non-null values, meaning that 1369 instances are missing this value. You can also already see that 37 features are numeric (38 if Id is not removed) and 43 of type object, which means they can hold any type of Python object. However since you loaded the data from a CSV file you can speculate they are text features. Also, when you look at the top five rows, you can notice that feature values of type object (e.g., MSZoning and Street) are repetitive, which means they are probably categorical features. You can find how many categories exist by using the Panda's `value_counts()` method. You can use Panda's `describe()` to show descriptive statitstics summary of the numerical features, such as mean. Notice that the `describe()` method excludes NaN values e.g, 'MasVnrArea' feature. See 1.2 for help

3 Discover and Visualize Dataset to Gain Insights

So far you have briely looked at the data to get a general understanding of the kind of data you are manupulating. To help you achieve indepth view of data, you will be doing some plotting as you explore the data. There are variety of ways to visualize data and gain insights, such as Histograms and Scatterplots. `Matplotlib.pyplot` module will be used for visualizations.

TASK 2. In your Python file, write code to check or plot relationships of features (remember to give names to title, xlable and ylable):

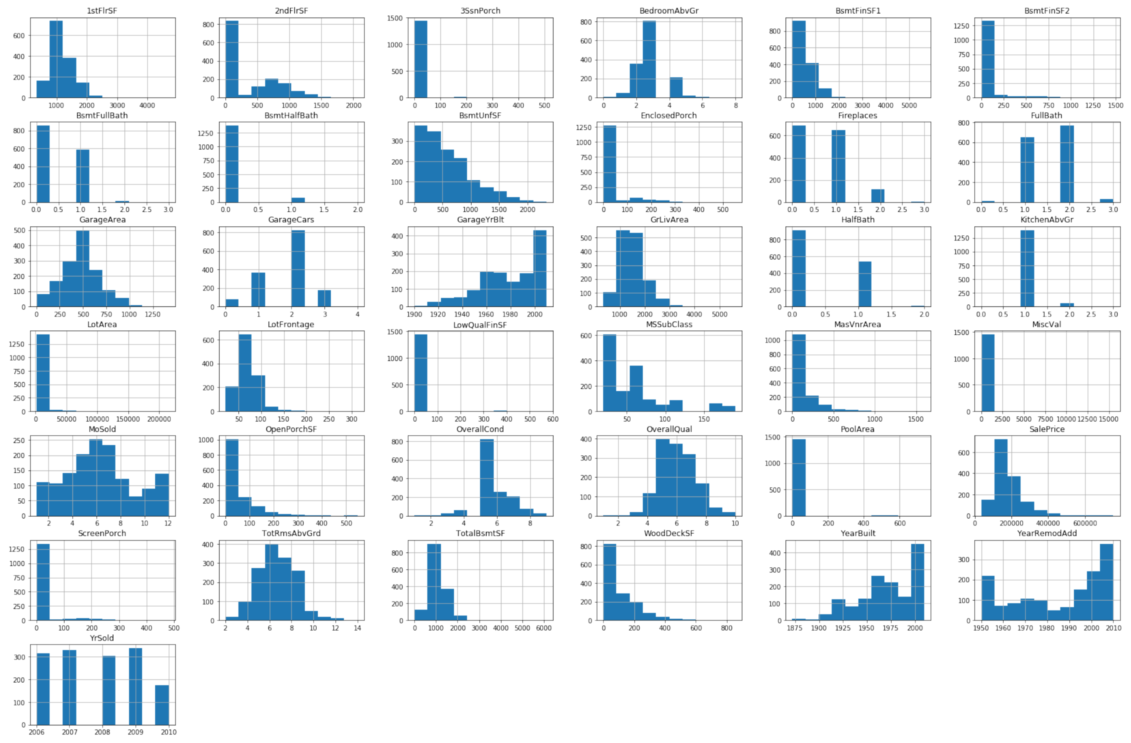
- Histogram of DataFrame using `DataFrame.hist()`
- Correlations of 'SalePrice' with others features using `DataFrame.corr()`
- Scatterplot to visualize correlations of few selected features ('SalePrice', 'OverallQual','GrLivArea', 'GarageArea', 'TotalBsmtSF', 'LotArea') using `DataFrame.scatter_matrix()`
- Scatterplot of 'SalePrice' and ' GrLivArea' using `plt.scatter()`
- Pivot table using `DataFrame.pivot_table()` between (a) SalePrice and SaleCondition, (b) SalePrice and Neighborhood, (c) SalePrice and MSZoning and SalePrice

Make sure you are able to inteprete the results and can give a brief explanation of the visualizations

3.1 Overview with Histograms

You can plot a histogram of each numerical feature to show number of instances for a given value range. `DataFrame.hist()` plots histogram over whole dataset, however in other cases you can choose to plot histogram of individual numerical feature.

Based on these histograms, take note that features have very different scales (x-axis) and that some features have 'tail-heavy' skewed ditribution. You will now proceed to explore relationship between features.



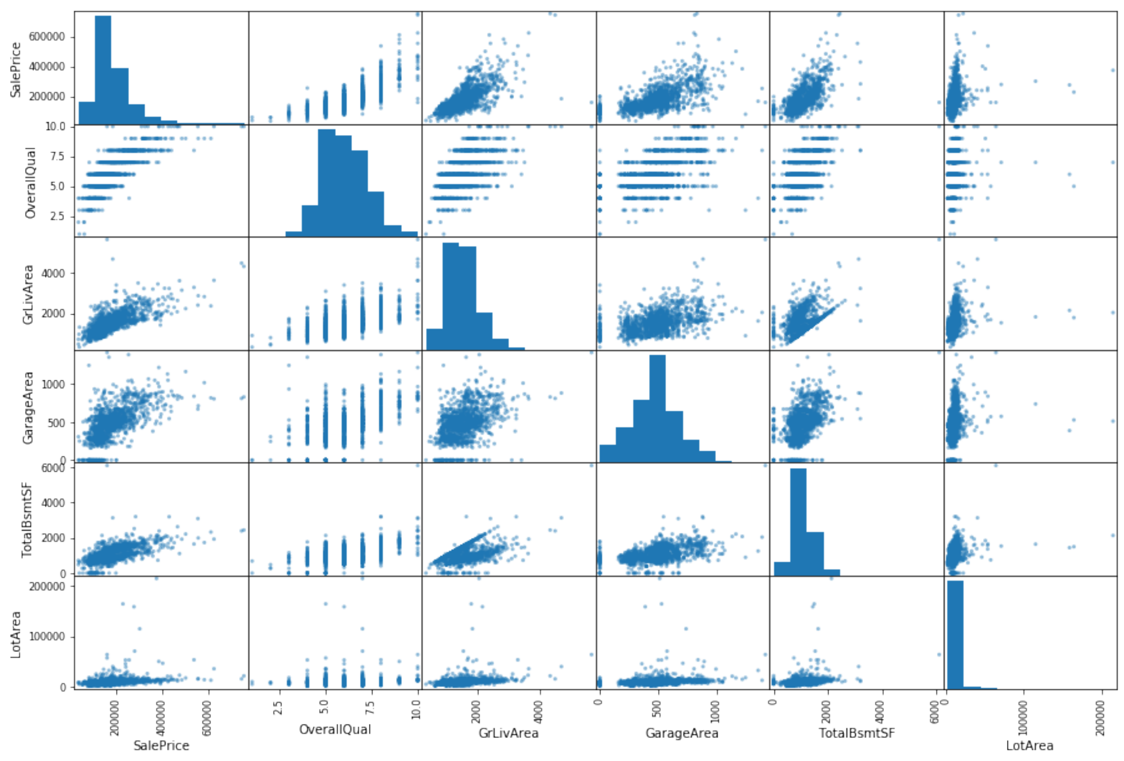
3.2 Looking for correlations

Since the dataset is not too large, you can compute correlation coefficient between every pair of features using Pandas `corr()` method. Correlations coefficients measures linear correlations and ranges from -1 to 1, whereby values close to 1 mean strong positive correlation and correlations close to zero mean there is no linear correlation.

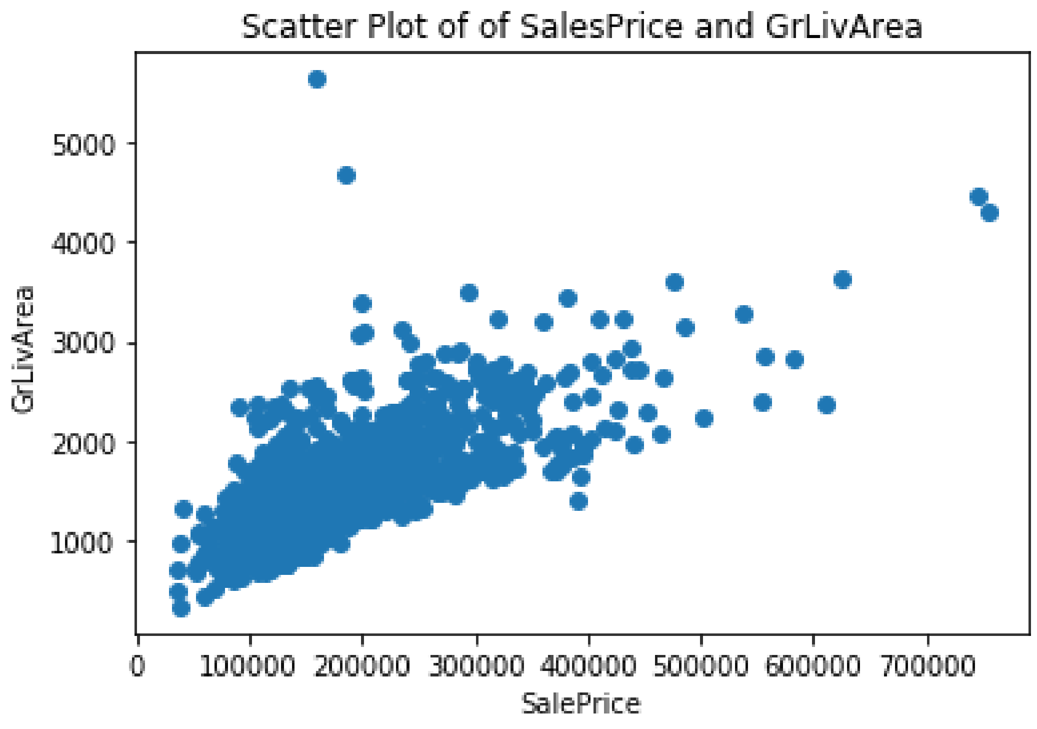
Using Panda's `corr()`, you need to examine the correlation between the features and the target 'SalePrice'. You will notice that the target 'SalePrice' variable is highly correlated with features like OverallQual, GrLivArea, GarageCars, GarageArea and TotalBsmtSF among others.

3.3 Overview with Scatterplots

You can use scatterplot to further explore data by checking the correlation between numeric features. Pandas' `pd.plotting.scatter_matrix()` plots every numerical feature against every other numerical feature. Since the dataset has several features you will only explore few promising features (with high correlation) namely: SalePrice, OverallQual, GrLivArea, GarageArea, TotalBsmtSF, and LotArea



From the scatterplot you can confirm strong positive correlations with the upward trends and notice that 'OverallQual' is a categorial feature. Among the most promising feature to predict 'SalePrice' is GrLivArea. Use `plt.scatter()` to further get detailed view of the correlations.



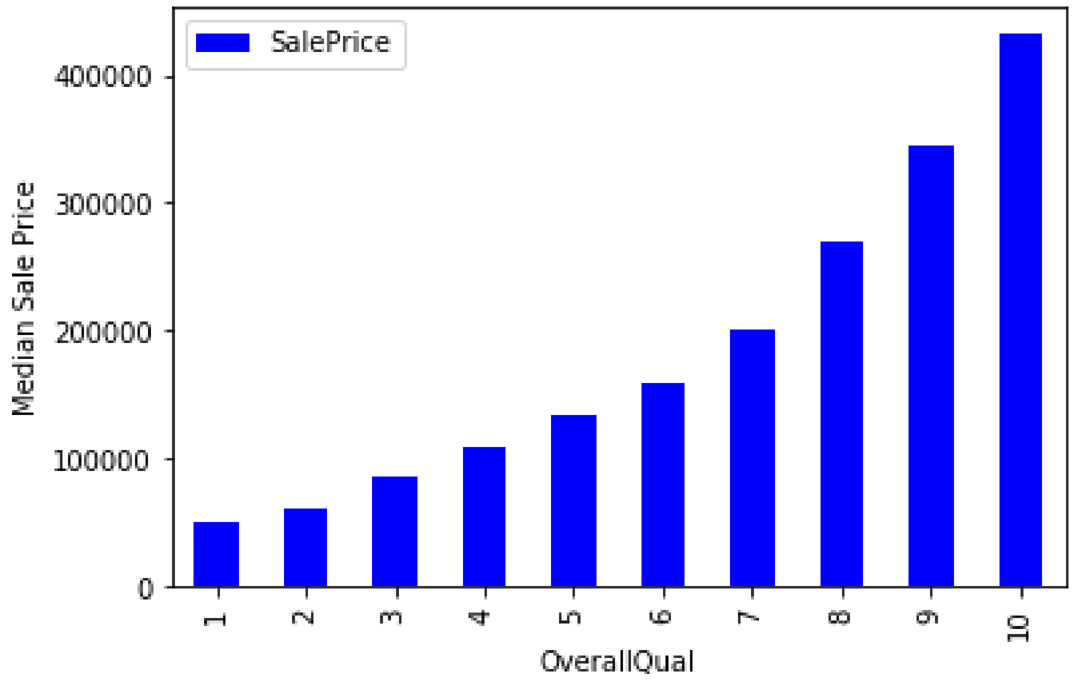
```
(1456, 80)
```

Based on correlation results, you have used scatterplot to further explore'GrLivArea' and 'SalePrice' features that revealed positive correlation through the upward trend. At the same time, outliers can be identified that show relatively low prices for large house sizes. Outliers are observed in houses with over 4000 GrLivArea as they contain either extremely high prices or low prices for such big area sizes.

These outliers need to be removed to prevent your ML algorithm from learning from them. Therefore, you will clean the data by removing outliers and only train on houses with less than 4000 GrLvArea. (Notice that often you will need to assess the impact of removing outliers prior to deciding their removal)

3.4 Relationships with Pivot Table

A pivot table is a table of statistics that summarizes the data of a more extensive table. In our data, a categorical feature 'OverallQual' has unique values in the interval of 1 to 10 inclusive. You can create a pivot table to investigate the relationship between this categorical feature (e.g., OverallQual) and SalePrice. This is accomplished with Panda's `pivot_table()` by setting index='OverallQual'and values='SalePrice' and choosing to look at the median. Use `Series.plot()` to visualize the pivot table more easily by creating the bar plot.



Exploration stage is an iterative process. It does not have to be absolute thorough at first but importantly is to gain intial insight to help you get a first reasonably good prototype and come back to this exploration step. For example, you may combine features and compute new correlations.

4 Prepare Dataset for Machine Learning Algorithm

The previous tasks have given you some insights into the dataset which you may opt to further prepare for ML algorithm. You were able to identify some oultiers that you removed and showed you interesting correlations between the target and the other features. Furthermore, while histograms showed tail-heavy distribution 'skew' requiring transformations, and transformations of non-numeric features have so far not been explored. You will write code to prepare dataset for ML algorithm.

TASK 3: Write code in your Python file to:

- 3.0 Separate features and the target variable using `DataFrame.drop('column', axis =1)` for features DataFrame object and `DataFrame['column'].copy()` for target DataFrame object. Use `train_data_features` and `train_data_target` as names of the new DataFrame objects.
- 3.1 Engineer numeric features (handle missing values and perform log transformation). See Task 3.1 for more details
- 3.2 Engineer categorical features (handle missing values, perform one-hot encoding). See Task 3.2 for more details

4.1 Separating features and target variable

First separate the target feature since we do not necessarily need to apply the same transformantion to predictors and target feature. You will use Panda's `drop('column', axis =1)` which creates a copy of the DataFrame and drops the specified column without affecting the original DataFrame.

```
Shape of features: (1456, 79)
Shape of target: (1456,)
```

4.2 Handling missing values

Missing values are blank spaces in our data table or as placeholder strings e.g., Not A Number (NaN). As most computational tools are unable to handle such missing values or would produce unpredictable results if we simply ignored them. Therefore, it is crucial that we take care of the missing values before we proceed with further analyses.

In this exercise you can accomplish handling of missing values by using Panda's `DataFrame.dropna()`, `DataFrame.drop()` and `DataFrame.dropfillna()`. To get an overview of missing values in current train dataset you will use the `isnull()` to return a DataFrame with Boolean values that indicate whether data is missing. The `sum()` can further be used to return the number of missing values per column.

You typically have three options to deal with missing values:

- Get rid of the instances with missing values with Panda's `dropna(])`

- Get rid of the whole feature with Panda's `drop()`
- Set the missing values to some value(zeros, mean, median, etc) with Panda' `fillna()`

Notice that, in real world deployments, since you cannot be sure there will not be any missing values in new data after training and deploying a ML model, you need to apply handling missing value to not just the current missing values. However in this exercise we only handle missing values in existing data.

5 Numeric Features (Feature Engineering)

As you start to consider features, you can check numeric features using Panda's `DataFrame.select_dtypes()` to return subset of columns matching the specified data types. To specify numeric datatype pass `include=[np.number]` which uses numpy datatype function.

You will process numeric features by first handling the missing values, and then handle skewness with for example log-transformation or perfom feature rescaling.

TASK 3.1 Write code in your Python file:

- Create a DataFrame object from `train_data_features` DataFrame object to contain numeric features `numeric_features`. You can use `DataFrame.select_dtypes()` and passing `include=[np.number]` to get numeric features.
- Check and handle missing values of numeric features. Check missing values in dataset using `DataFrame.isnull().sum().sort_values(ascending=False)`. See 3.1.1 handling missing values of numeric features for implementation details
- Engineer the numeric features. See 3.1.2. Processing of numeric features for implementation details.

5.1 Handling missing values of numeric features

The dataset contains missing values for 'LotFrontage', 'GarageYrBlt' and 'MasVnrArea. You will proceed to handle the missing values in the following ways:

- LotFrontage: fill median value (sort data by 'Neighborhood' before imputing median). Use Panda's `DataFrame.groupby()` and `DataFrame.transform()`
- MasVnrArea: Fill 0, since NA means 'no masonry veneer'
- GarageYrBlt: Fill 0, since NA means 'no garage'

5.2 Processing of numeric features (log-transform)

When performing regression, sometimes it is good to apply log-transformation to the target variable when it is skewed. Important to notice that the predictions generated by the final model will also be log-transformed, so you will need to convert the predictions back to their orignial form later.

You will use numpy's log-transformation function `np.log()` and pass DataFrame object to apply the transformation. You will apply log-transformation to target 'SalePrice' variable. You can use `np.exp()` to reverse the transformation. Check for skeweness before and after log-transformation using Panda's `skew()` function specifically DataFrame.SalePrice.skew()

```
Skew (no log transform): 1.5659592925562151
Skew (log transform): 0.06544882031646646
Expected shape of features after numeric FE: (1456, 79)
Expected shape of target after numeric FE: (1456,)
```

6 Categorical Features (Feature Engineering)

Since most machine learning algorithms prefer to work with numbers, you will covert the categories from text to numbers. To do the latter, you will use Pandas'one-hot encoding `get_dummies()`. You will first need to handle missing data of categorical features. Often it is better to gather domain knowledge in order to make the best decision when dealing with missing data of categorical features. In our exercise the Ames Housing documentation can help in understanding the missing values. For example, *PoolQC* features the value is NaN when PoolArea is 0, meaning there is no Pool.

TASK 3.2 Write code in your Python file to:

- Handle numeric features that were deemed categorical by changing their dttype to object using `DataFrame[col].apply(str)`
- Create a DataFrame object from `train_data_features` DataFrame object to contain categorical features `numeric_features`. Use `DataFrame.select_dtypes()` and passing `exclude=[np.number]` to get categorical features. Similar to first part of Task 3.1
- Check and handle missing values of categorical features. Check missing values in dataset using `DataFrame.isnull().sum().sort_values(ascending=False)`. Handle missing values for categorical features (PoolQC,MiscFeature, Alley, Fence, FireplaceQu, GarageCond, GarageType, GarageFinish, GarageQual,BsmtExposure, BsmtFinType2, BsmtCond, BsmtQual, MasVnrType) by filling 'None'. You remove the one observation with missing value for Electrical feature.
- Perform one-hot encoding for categorical features. See 3.2.2. Processing of categorical features (one-hot encoding) for implementation details.

6.1 Handling missing values of categorical features

Several categorical features have missing value. You will handle values of text and categorical as follows as noted in the data description:

- PoolQC: NA means 'no pool', thus fill "None"
- MiscFeature: NA means 'no misc feature', thus fill "None"
- Alley: NA mean 'no alley access', thus fill "None"
- Fence: NA means 'no fence', thus fill "None"
- FireplaceQu: NA means 'no fireplace', thus fill "None"

- GarageCond, GarageType, GarageFinish and GarageQual: fill "None"
- BsmtExposure, BsmtFinType2, BsmtCond, BsmtQual: fill "None", since NaN means no basement
- Electrical: Only one observation with NAN, you can ignore or remove observation with missing value, using `dropna(subset=['column'])`. If you choose to remove the observation you need to do it before separating features and target

6.2 Processing of categorical features (one-hot encoding)

For categorical features, you may want to use **one-hot encoding** to make transformations useful for building the model. You can do this by using Panda's `DataFrame.get_dummies()`.

When using one-hot encoding, it is important to remember that for a number of categories of a categorical feature you will create a set of new features equal to the number of its categories. In order not to increase the number of new features, you will perform one-hot encoding on selected categorical features. Furthermore, some categorical features may contain information in their ordering set.

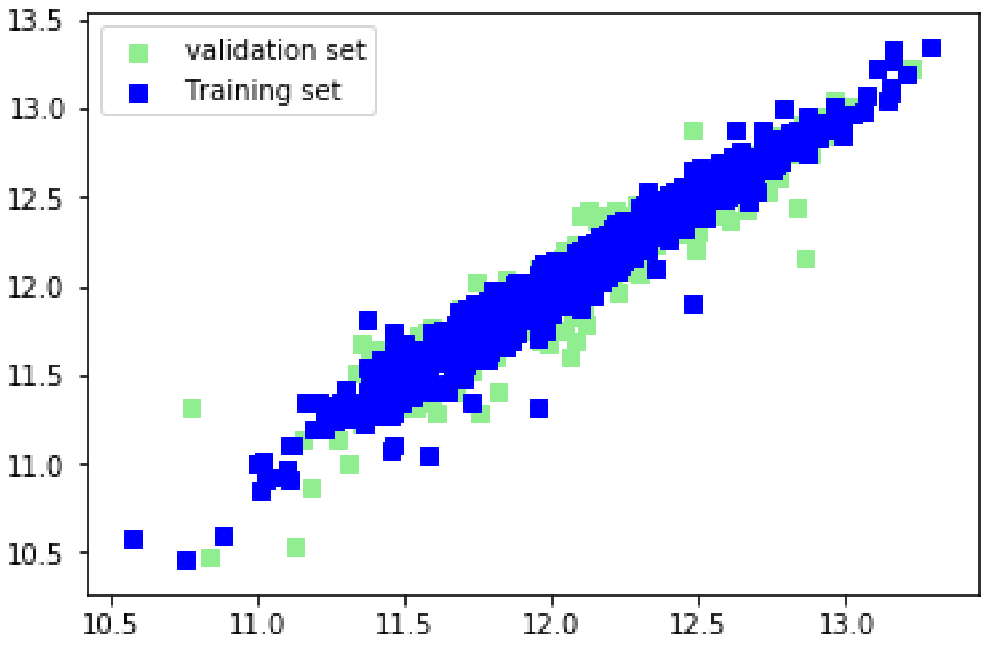
```
(1456, 332)
Expected shape of features after numeric FE: (1456, 332)
Expected shape of target after numeric FE: (1456,)
```

7 Build and Evaluate a simple linear model

In this part of the exercise you will build a simple linear model. First use `train_test_split()` from sklearn to create training set and testing set. The train_test_split() returns four objects: - X_train, a subset of features used for training - X_test, a subset of features which will be test set - y_train, the target variable that corresponds to X_train - y_test, the target variable that corresponds to X_test

Next, you will sent random_state=0 that provides reproduciable results. The test_size parameter tells the function what proportion of the data should be in the test partition. In this exercise, about 20% of the data is devoted to the test set.

To build a linear model, you will first instantiate the model and then fit the model. You will fit the model using features_train and target_train and then score/evaluate them using features_test and target_test. The `lr.fit()` will fit a linear regression on the features and target variable that you give.



Task 4: Interprete model evaluation results Evaluate the results of your model with root mean square error (RMSE) from sklearn metrics.meas_squared_error()

The resulting simple linear model seems is fitting perfectly to data, and due to overfitting it will may not perfectly perform well with new unseen data. You can work to further improve the model by engineering the features, for example by training on few but high predictors. You can also use other ML algorithms to perform a similar task.