What is the problem?

Let’s get started to make a prediction on our first machine learning algorithm with a rich dataset on housing prices from Ames, Iowa. Each row in the dataset describes the properties of a single house as well as the amount it was sold for. In this concept, we'll build models that predict the final sale price of a house based on its other attributes. The original data set contains 82 features and 2930 data points.

However, for the purpose of understanding how Linear Regression works, we will specifically, work on the following features of the house

- ExterQual
- AllFlrsSF
- GrLivArea
- SimplOverallCond
- GarageArea
- TotRmsAbvGrd
- LotFrontage

Brief explanation of the dataset & features

- `ExterQual (Ordinal)`: Evaluates the quality of the material on the exterior
  5: Excellent 4: Good 3: Average/Typical 2: Fair 1: Poor
- `AllFlrsSF(Continuous)`: Total square feet for 1st and 2nd floor combined
- `GrLivArea (Continuous)`: Above grade (ground) living area square feet
- `SimplOverallCond (Ordinal)`: Rates the overall condition of the house
  1: Bad 2: Average 3: Good
- `Garage Area (Continuous)`: Size of garage in square feet
- `TotRmsAbvGrd (Nominal)`: Total rooms above grade (does not include bathrooms)
- `LotFrontage (Continuous)`: Linear feet of street-connected to property

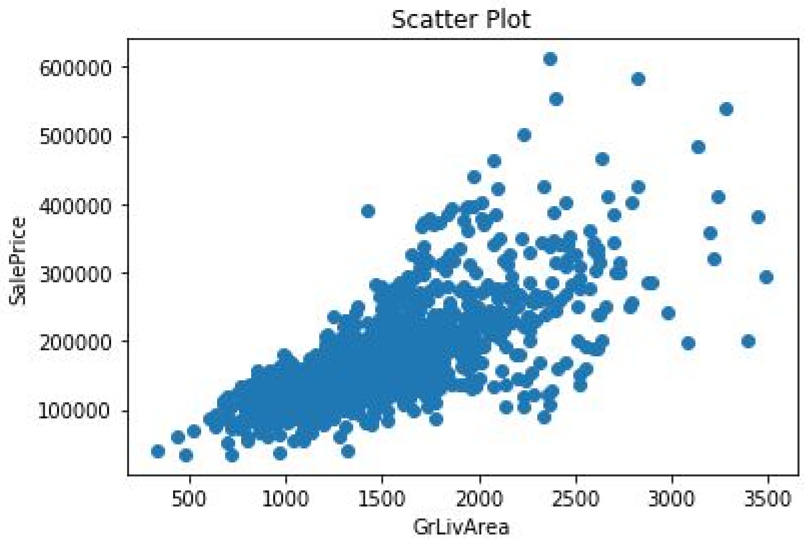
What do we want as an outcome?

Using the set of some basic attributes that are related to the price of the house, predict the sale price for a new house using Linear Regression.

Need for Linear Regression

Intuition for Linear regression

Let's start with what information we have: The main goal is to build a machine learning model that can predict the selling price of the house given some of its features like `GrLivArea`, `Garage Area` etc. If you do a scatter plot with `SalePrice` which is the target variable and `GrLivArea`, a feature of the house, we might get something similar to the following:



Know Your Linear Regression

In simple linear regression, we establish a relationship between the target variable and input variables by fitting a line, known as the regression line.

In general, a line can be represented by linear equation $y=mx+b$. Where y is the dependent variable, x is the independent variable, m is the slope, b is the intercept.

In machine learning, we rewrite our equation as $y(x) = \theta_0 + \theta_1 x$ where θ_is are the parameters of the model, x is the input, and y is the target variable. This is the standard notation in machine learning and makes it easier to add more dimensions. We can simply add variables $\theta_2, \theta_3, \dots, \theta_n$ and $x_2, x_3, \dots$ as we add more dimensions. Different values of θ_0 and θ_1 will give us different lines of fit.

Each of the values of the parameters determine what predictions the model will make. For example, let's consider $(\theta_0, \theta_1) = (0.0, 0.2)$, and the first data point, where $x = 3456$ and $y_{true} = 600$. The prediction made by the model, $y(x) = 0.0 + 0.2 * 3456 = 691.2$. *If instead the weights were* $(\theta_0, \theta_1) = (80.0, 0.15)$ *, then the prediction would be* 3456 = 598.4y(x) = 80.0 + 0.15 * 3456 = 598.4, which is much closer to the $y_{true} = 600$.

This difference between the actual value of the target and the predicted value is called the **residual**. You should always select the line with the least residual as the line fits the data points appropriately. To put it more formally the residual $\varepsilon = y_{true} - y_{pred}$. Or in other words, $y_{true} = y_{pred} + \varepsilon$. Further, the residual could be modeled as the sum of the individual residual errors. You can interpret these errors as the individual errors due to wrong parameter estimation or error in the measurement of features.

$$y_{true} = y_{pred} + \varepsilon = \theta_0 + \theta_1 x + \varepsilon$$

Why Linear Regression for this data

We observed a linear relationship between the features to the price of the house (which is a continuous real-valued number), so it is a regression task. So, `Linear Regression` will do a good job in predicting our target i.e. **Sale Price**.

In our dataset the linear relationship between the features and target will be captured in the following form: $SalesPrice = \theta_0 + \theta_1 * ExterQual + \theta_2 * AllFlrsSF + \theta_3 * GarageArea + \theta_4 * SimplOverallCond + \theta_5 * GrLivArea + \theta_6 * TotRmsAbvGrd + \theta_7 * LotFrontage + \varepsilon$, where ε corresponds to the residual.

Instead of using the full feature names, let's use variables $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_7$, where $\mathbf{x}_i$ corresponds to the column vector containing the feature value of all houses in the dataset. $\mathbf{y}$ corresponds to the Sales price of all houses in the dataset. Now the equation transforms to

$$\mathbf{y} = \theta_0 + \theta_1 * \mathbf{x}_1 + \theta_2 * \mathbf{x}_2 + \theta_3 * \mathbf{x}_3 + \theta_4 * \mathbf{x}_4 + \theta_5 * \mathbf{x}_5 + \theta_6 * \mathbf{x}_6 + \theta_7 * \mathbf{x}_7 +$$

In the matrix form, it becomes

$$\mathbf{y} = \begin{bmatrix} 1 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_6 & \mathbf{x}_7 \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \end{bmatrix}$$

Here, our main goal is to find the values of the coefficients $\theta_0, \theta_1, \cdots, \theta_7$ to find the best fit line representing our data points.

Linear Relationship

According to this assumption, the relationship between response (Dependent Variables) and feature variables (Independent Variables) should be linear.

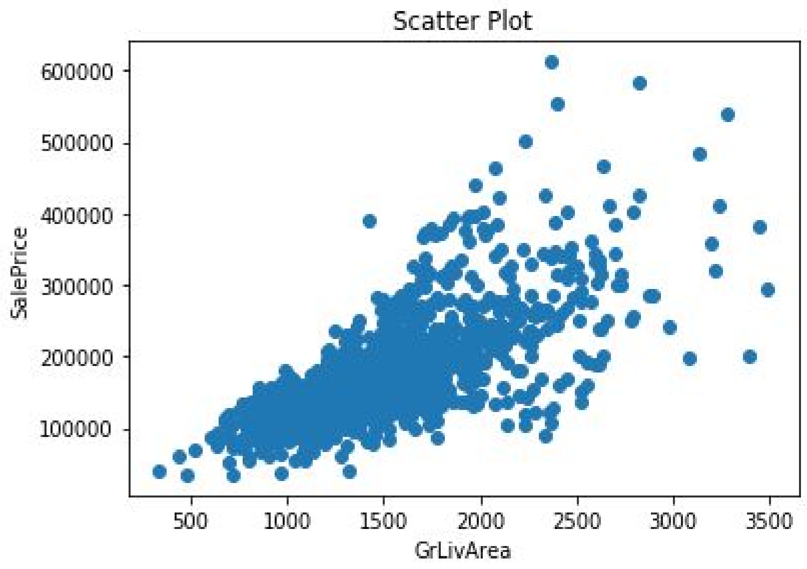
Why it is important?

- Linear regression only captures the linear relationship, as it is trying to fit a linear model to the data.

How to validate it?

- The linearity assumption can be tested using scatter plots.

In the scatter plot for `SalePrice` vs `GrLivArea`, you can clearly see that a linear pattern is evident here i.e. as the value of `GrLivArea` increases the `SalePrice` also increases, and vice-versa.



Little or No Multicollinearity Assumption

What is multicollinearity?

It is assumed that there is little or no multicollinearity in the data. But what do we mean by multicollinearity? Well, multicollinearity occurs when independent variables in a regression model are correlated. This correlation is a problem because independent variables should be independent. If the degree of correlation between variables is high enough, it can cause problems when you fit the model and interpret the results.

Why sweat over multicollinearity?

The interpretation of a regression coefficient is that it represents the mean change in the dependent variable for each unit change in an independent variable when you hold all of the other independent variables constant. However, when independent variables are correlated, changes in one variable, in turn, shifts another variable/variable. The stronger the correlation, the more difficult it is to change one variable without changing another. It becomes difficult for the model to estimate the relationship between each independent variable and the dependent variable independently because the independent variables tend to change in unison.

Effects of multicollinearity

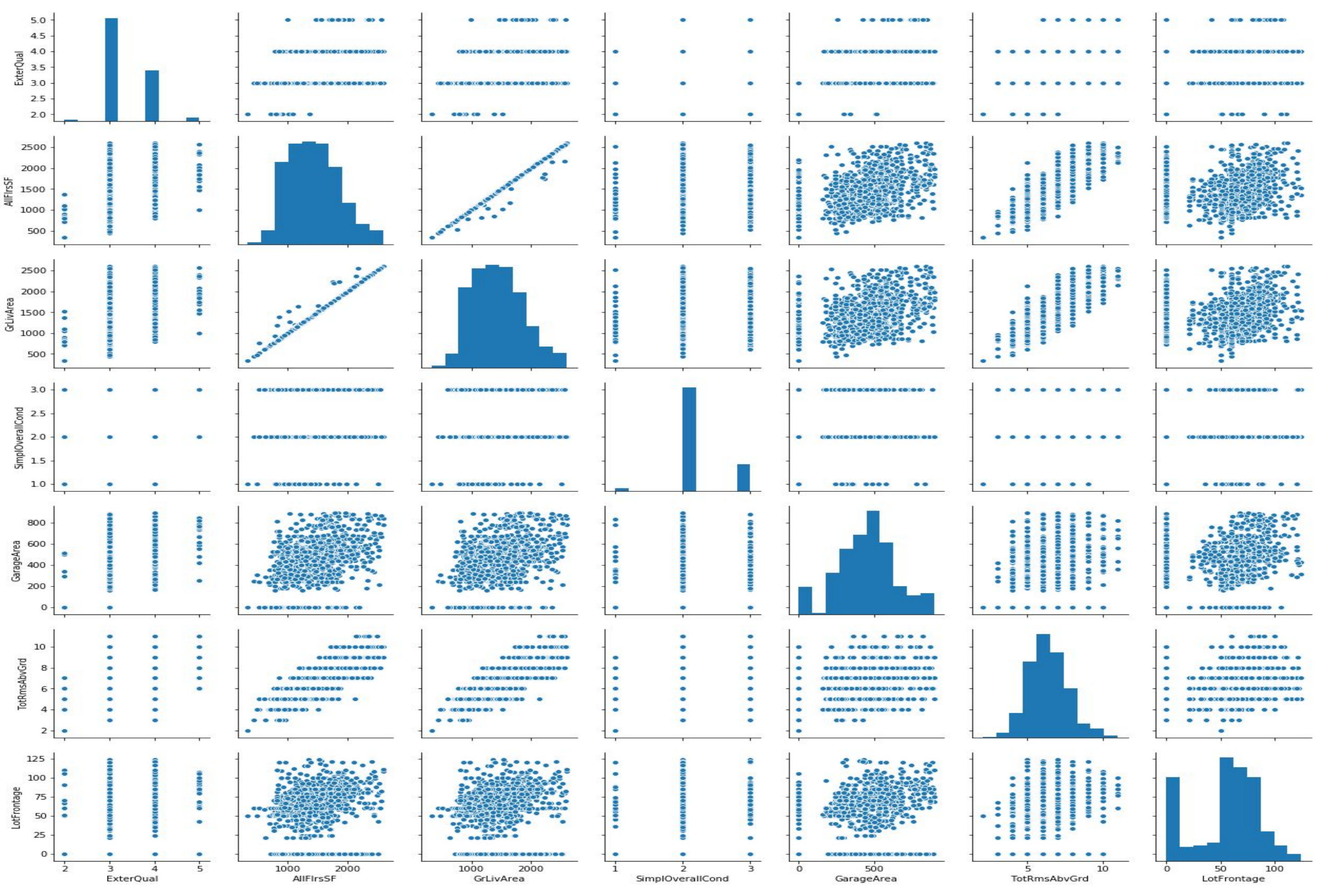
- It results in unstable parameter estimates which makes it very difficult to assess the effect of independent variables.
- Weakens the statistical power of the regression model

How to validate it?

- Multicollinearity occurs when the features (or independent variables) are not independent of each other. Pair plots of features help validate.
- You can also calculate the correlation coefficient (Pearson or Spearman) to figure out which features are correlated.

Treating multicollinearity

- Remove some of the highly correlated independent variables.
- Linearly combine the independent variables, such as adding them together.



Homoscedasticity Assumption

Homoscedasticity describes a situation in which the error term (that is, the "noise" or random disturbance in the relationship between the independent variables and the dependent variable) is the same across all values of the independent variables.

Why it is important:

- Generally, non-constant variance arises in the presence of outliers or extreme leverage values.

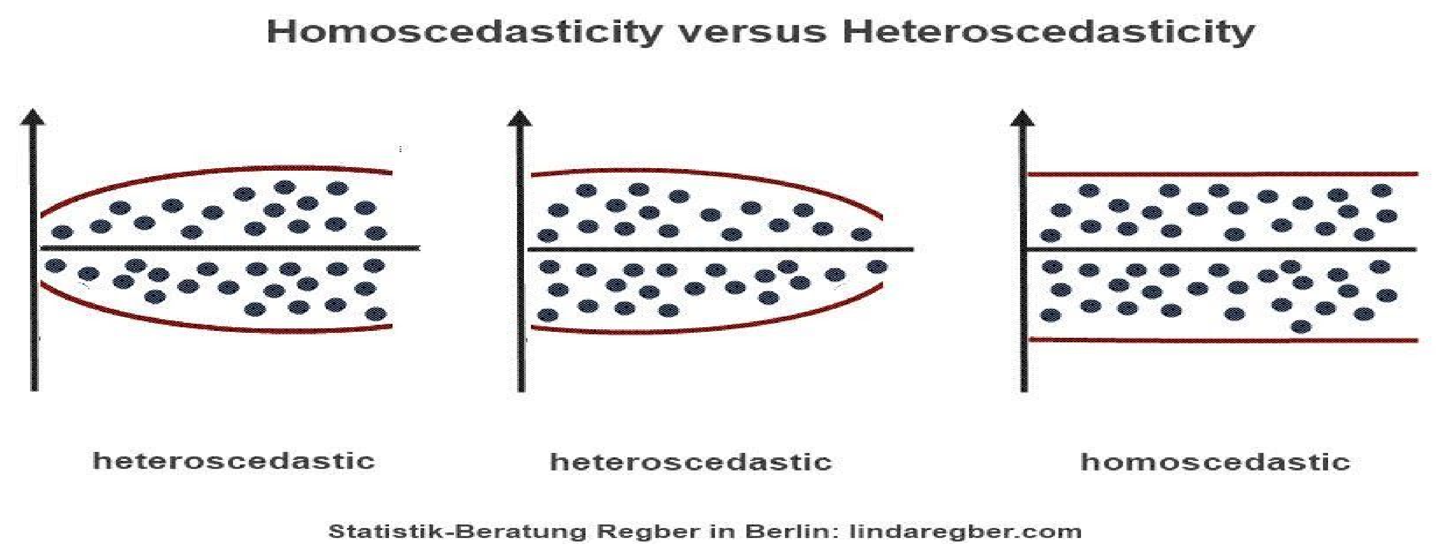
How to validate:

- The plot between error (residuals) vs predicted/fitted values. If there is no fan-shaped pattern visible, then it satisfies this assumption.

In the image below, shown are three different plots for residuals (True value - Predicted value) and Predicted value. Let's discuss these plots in detail:

- The left two plots are instances of heteroscedasticity where the variance either increases (left plot) or decreases (right plot). So, it violates the assumption of constant variance.
- The rightmost plot is an instance that satisfies the assumption of constant variance among the residuals.

The residual plot i.e. Residuals vs Predicted value plot should aspire to achieve the third plot for linear regression.


Homoscedasticity versus Heteroscedasticity
heteroscedastic
heteroscedastic
homoscedastic
Statistik-Beratung Regber in Berlin: lindaregber.com

Little or No autocorrelation in residuals

There should be little or no autocorrelation in the data. Autocorrelation occurs when the residual errors are not independent of each other.

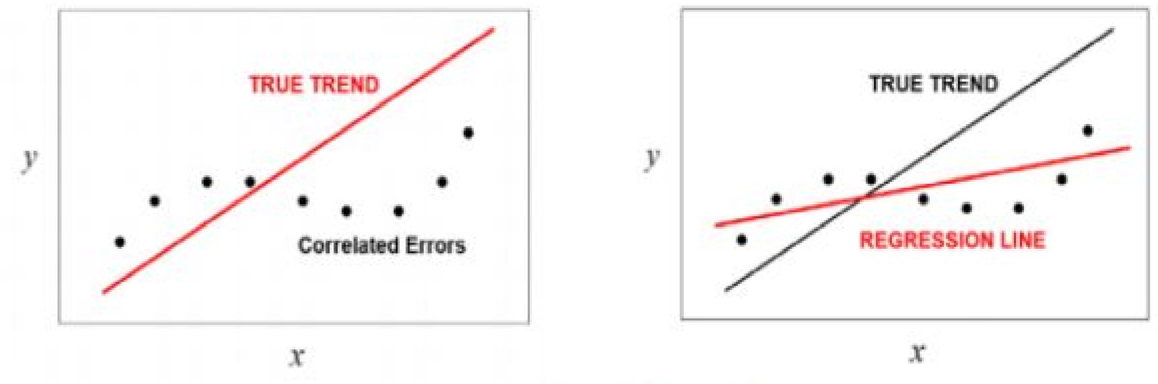
Why it is important:

- The presence of correlation in error terms drastically reduces the model's accuracy. This usually occurs in time series models. If the error terms are correlated, the estimated standard errors tend to underestimate the true standard error.

How to validate:

- Residual vs Time plot: Look for the seasonal or correlated pattern in residual values.

In the plot, two plots are shown where the regression line (right plot) deviates from the true trend. Hence, it is necessary to take care of it.



Source: Penn Engineering

Normal Distribution of error terms

A common misconception about linear regression is that it assumes that the outcome Y is normally distributed. Actually, linear regression assumes normality for the residual errors ε, which represent variation in Y not explained by the predictors.

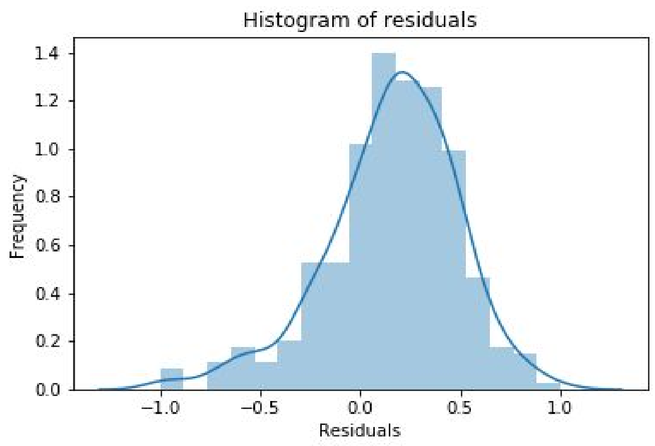
Why it is important:

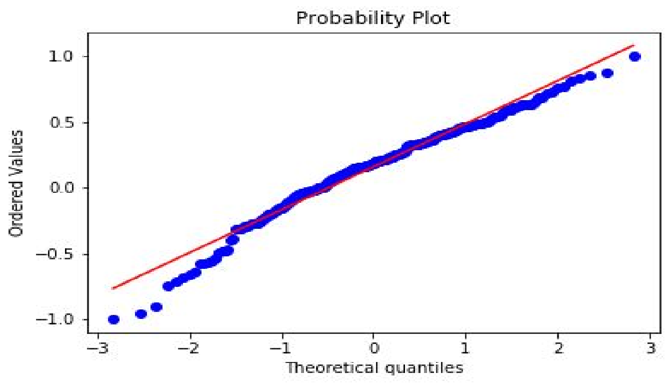
- Due to the Central Limit Theorem, we may assume that there are lots of underlying facts affecting the process and the sum of these individual errors will tend to behave like in a zero-mean normal distribution. In practice, it seems to be so.

How to validate:

- You can look at the QQ plot of the residuals. QQ plot is a scatterplot created by plotting two sets of quantiles against one another. For our case, we plot the residual quantiles against the theoretical quantiles corresponding to a normal distribution. If the QQ plot is a straight line, then we can infer that the errors follow a normal distribution.
- You should observe a normal curve on plotting a histogram of the residuals.

The following graphs were taken after we have calculated the residuals and made a Q-Q plot and a histogram for residuals. Observing them we can tell that they are very close to satisfying the normal distribution assumption.

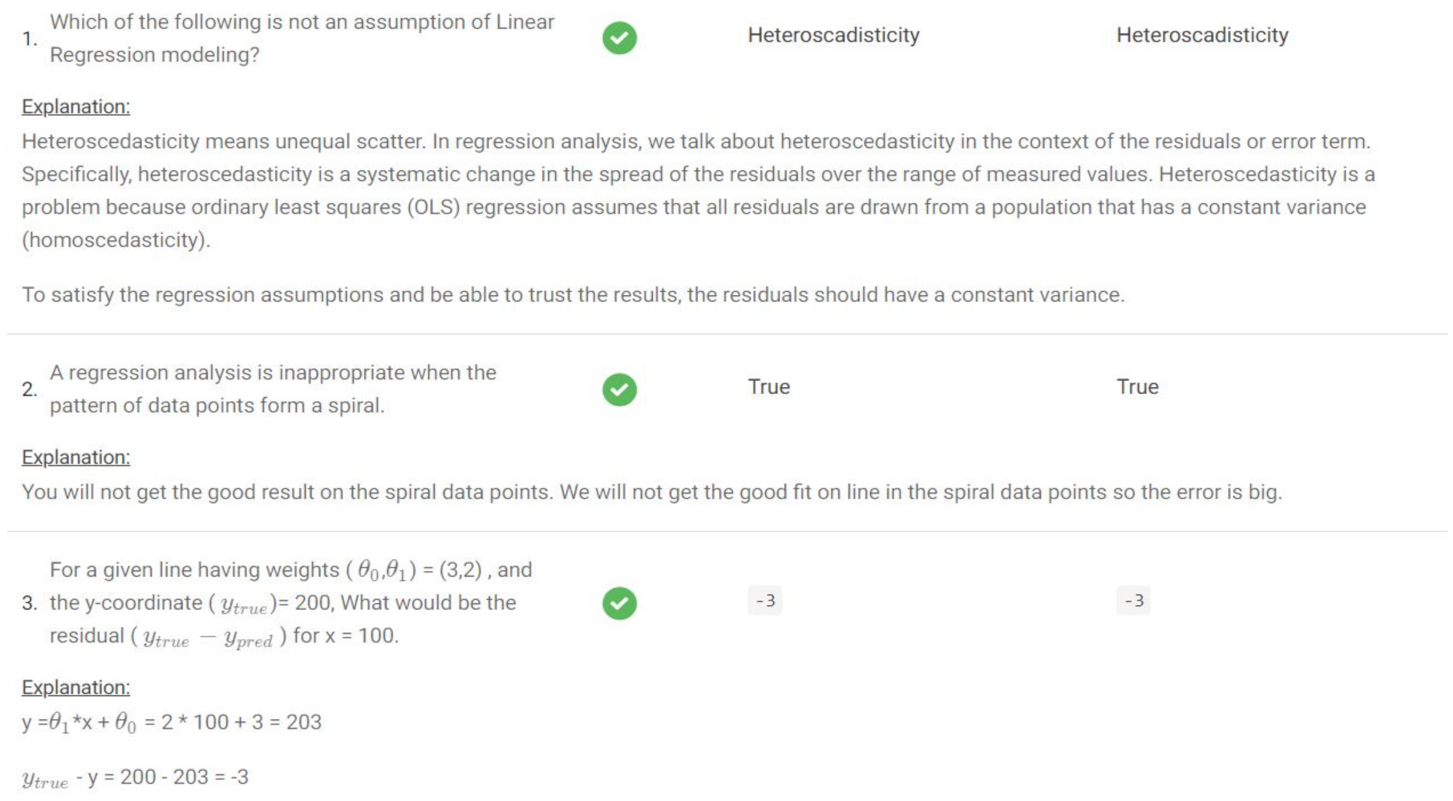




Why you should care about these assumptions?

In a nutshell, your linear model should produce residuals that have constant variance and are normally distributed, features are not correlated with themselves or other features, etc. If these assumptions hold true, the OLS procedure (discussed in the next chapter) creates the best possible estimates for the coefficients of linear regression.

Another benefit of satisfying these assumptions is that as the sample size increases to infinity, the coefficient estimates converge on the actual population parameters.



1.	Which of the following is not an assumption of Linear Regression modeling?	✓	Heteroscadisticity	Heteroscadisticity

Explanation:

Heteroscedasticity means unequal scatter. In regression analysis, we talk about heteroscedasticity in the context of the residuals or error term. Specifically, heteroscedasticity is a systematic change in the spread of the residuals over the range of measured values. Heteroscedasticity is a problem because ordinary least squares (OLS) regression assumes that all residuals are drawn from a population that has a constant variance (homoscedasticity).

To satisfy the regression assumptions and be able to trust the results, the residuals should have a constant variance.

2.	A regression analysis is inappropriate when the pattern of data points form a spiral.	✓	True	True

Explanation:

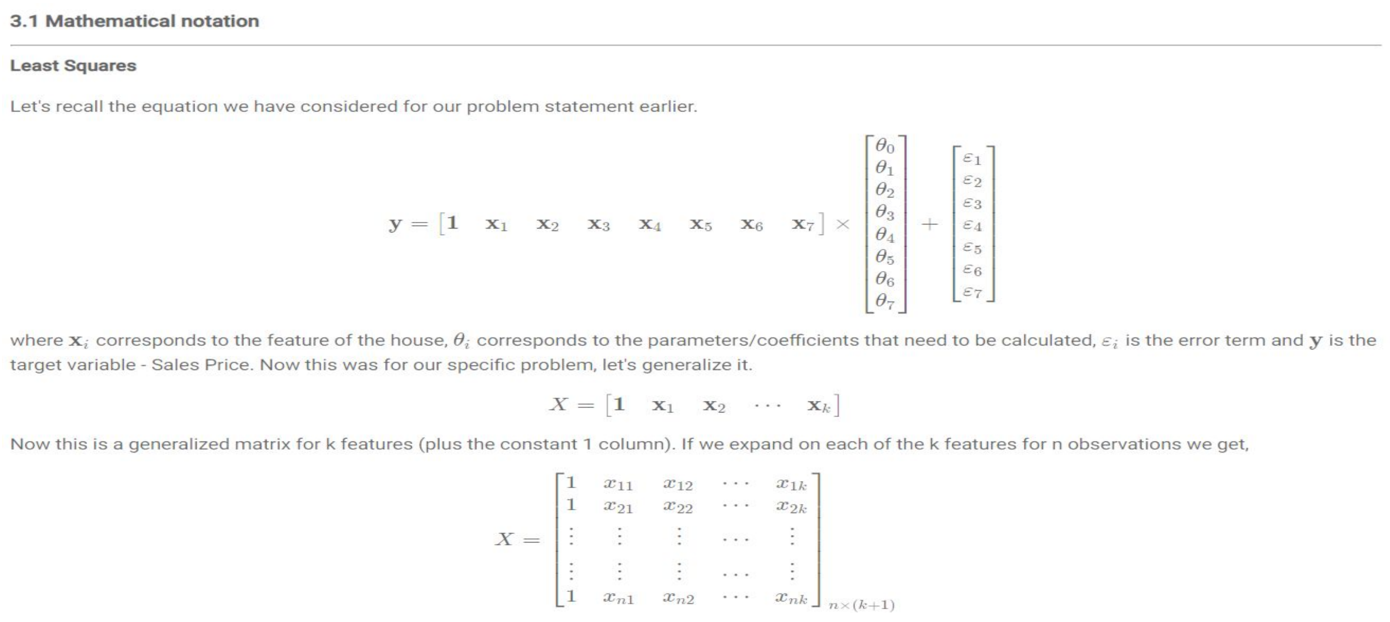
You will not get the good result on the spiral data points. We will not get the good fit on line in the spiral data points so the error is big.

3.	For a given line having weights (θ_0,θ_1) = (3,2) , and the y-coordinate (y_{true})= 200, What would be the residual ($y_{true} - y_{pred}$) for x = 100.	✓	-3	-3

Explanation:

y =θ_1*x + θ_0 = 2 * 100 + 3 = 203

y_{true} - y = 200 - 203 = -3



3.1 Mathematical notation

Least Squares

Let's recall the equation we have considered for our problem statement earlier.

$$\mathbf{y} = \begin{bmatrix} \mathbf{1} & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_6 & \mathbf{x}_7 \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \end{bmatrix}$$

where $\mathbf{x}_i$ corresponds to the feature of the house, θ_i corresponds to the parameters/coefficients that need to be calculated, ε_i is the error term and $\mathbf{y}$ is the target variable - Sales Price. Now this was for our specific problem, let's generalize it.

$$X = \begin{bmatrix} \mathbf{1} & \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k \end{bmatrix}$$

Now this is a generalized matrix for k features (plus the constant 1 column). If we expand on each of the k features for n observations we get,

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}_{n\times(k+1)}$$

which is a $n \times (k+1)$ matrix (k+1 to accomodate for column of 1).

Then $\mathbf{y}$ can be generalized to $n \times 1$ vector of observations on the target variable.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}_{n\times 1}$$

Let ε be the error term associated with each observation.

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \vdots \\ \varepsilon_n \end{bmatrix}_{n\times 1}$$

Let θ be the coefficient of each of the features. These are the parameters that need to be estimated by the model and there would be k parameters for k features.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \vdots \\ \theta_k \end{bmatrix}_{(k+1)\times 1}$$

Combining all these we get,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}_{n\times(k+1)} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \vdots \\ \theta_k \end{bmatrix}_{(k+1)\times 1} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \vdots \\ \varepsilon_n \end{bmatrix}_{n\times 1}$$

Or in a compressed form we can express it as

$$\mathbf{y} = X\theta + \varepsilon$$

Our goal is to estimate the parameters in the θ vector. The vector of residuals is given by $\varepsilon = \mathbf{y} - X\theta$. If we take the actual value of the residuals, they might be negative. To avoid that, we take the squared sum of the residuals.

$$\varepsilon_0^2 + \varepsilon_1^2 + \cdots + \varepsilon_n^2 = \begin{bmatrix} \varepsilon_0 & \varepsilon_1 & \cdots & \varepsilon_n \end{bmatrix} \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \varepsilon^T \varepsilon$$

$\varepsilon^T \varepsilon$ is also known as **cost function** which we want to minimize.

Substituting $\varepsilon = \mathbf{y} - X\theta$ we get

$$\varepsilon^T \varepsilon = (\mathbf{y} - X\theta)^T (\mathbf{y} - X\theta)$$
$$\varepsilon^T \varepsilon = (\mathbf{y}^T - (X\theta)^T)(\mathbf{y} - X\theta)$$
$$\varepsilon^T \varepsilon = (\mathbf{y}^T - \theta^T X^T)(\mathbf{y} - X\theta)$$
$$\varepsilon^T \varepsilon = \mathbf{y}^T \mathbf{y} - \mathbf{y}^T X\theta - \theta^T X^T \mathbf{y} + \theta^T X^T X\theta$$
$$\varepsilon^T \varepsilon = \mathbf{y}^T \mathbf{y} - 2\theta^T X^T \mathbf{y} + \theta^T X^T X\theta$$

$\mathbf{y}^T X\theta = \theta^T X^T \mathbf{y}$ because the transpose of a scalar is a scalar.

Now we need to find the θ that minimizes the sum of squared residuals. (That is why the name - Ordinary Least Squares). To find the θ, we take derivative w.r.t θ.

$$\frac{\partial \varepsilon^T \varepsilon}{\partial \theta} = -2X^T \mathbf{y} + 2X^T X\theta = 0$$
$$(X^T X)\theta = X^T \mathbf{y}$$

$X^T X$ is a square matrix $(k + 1 \times k + 1)$ and it is also symmetric. By multiplying $(X^T X)^{-1}$ both sides, we get

$$(X^T X)^{-1}(X^T X)\theta = (X^T X)^{-1} X^T \mathbf{y}$$
$$\theta = (X^T X)^{-1} X^T \mathbf{y}$$

Let's consider a bivariate case. Here we have a single feature $\mathbf{x}$ which influences a target variable $\mathbf{y}$. Let's consider n observations of the same. Then our linear regression model is $\mathbf{y} = \theta_0 + \theta_1 \mathbf{x}$. In the matrix form, it becomes

$$\mathbf{y} = X\theta + \varepsilon$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Solving using $\theta = (X^T X)^{-1} X^T \mathbf{y}$, we get

$$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \bar{y} - \theta_1 \bar{x} \\ \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \end{bmatrix}$$

You can simplify representing the 2nd term as follows:

$$\theta_1 = \frac{SS_{xy}}{SS_{xx}}$$

where SS_{xy} is the sum of cross-deviations of y and x:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

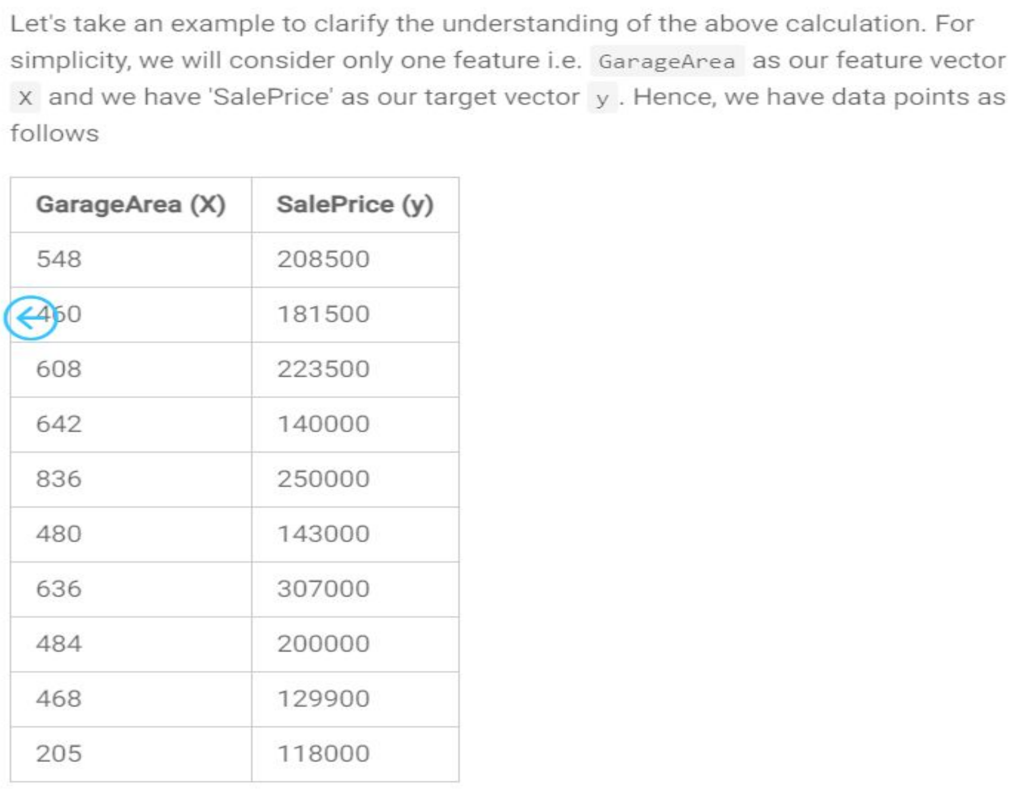
and SS_{xx} is the sum of squared deviations of x:

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

Ordinary Least Squates

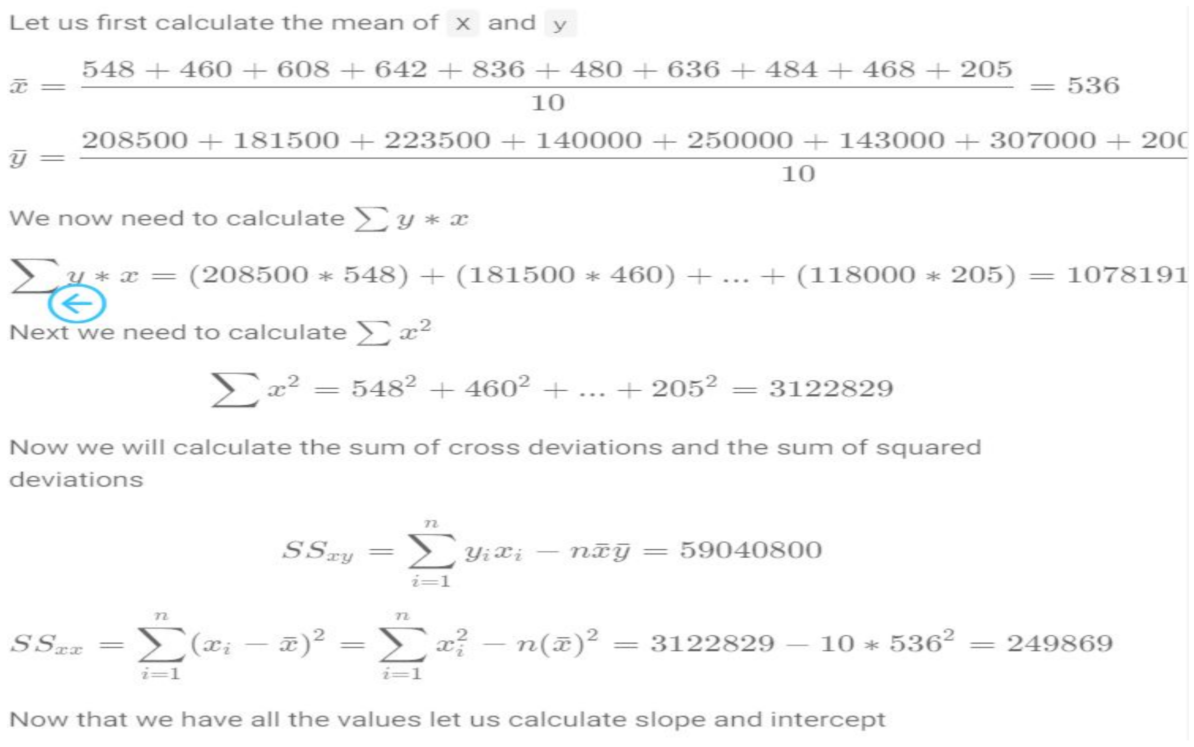
The ordinary least squares (OLS) approach to regression allows us to estimate the parameters of a linear model. The goal of this method is to determine the linear model that minimizes the sum of the squared errors between the observations in a dataset and those predicted by the model.

Discrete example to understand OLS



Let's take an example to clarify the understanding of the above calculation. For simplicity, we will consider only one feature i.e. `GarageArea` as our feature vector `X` and we have 'SalePrice' as our target vector `y`. Hence, we have data points as follows

GarageArea (X)	SalePrice (y)
548	208500
460	181500
608	223500
642	140000
836	250000
480	143000
636	307000
484	200000
468	129900
205	118000



Let us first calculate the mean of `X` and `y`

$$\bar{x} = \frac{548 + 460 + 608 + 642 + 836 + 480 + 636 + 484 + 468 + 205}{10} = 536$$

$$\bar{y} = \frac{208500 + 181500 + 223500 + 140000 + 250000 + 143000 + 307000 + 20(}{10}$$

We now need to calculate $\sum y * x$

$$\sum y * x = (208500 * 548) + (181500 * 460) + \ldots + (118000 * 205) = 1078191$$

Next we need to calculate $\sum x^2$

$$\sum x^2 = 548^2 + 460^2 + \ldots + 205^2 = 3122829$$

Now we will calculate the sum of cross deviations and the sum of squared deviations

$$SS_{xy} = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y} = 59040800$$

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2 = 3122829 - 10 * 536^2 = 249869$$

Now that we have all the values let us calculate slope and intercept

$$slope = \theta_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{59040800}{249869} = 236$$

$$intercept = \theta_0 = \bar{y} - \theta_1 \bar{x} = 190140 - 236 * 536 = 63644$$

So now if we want to predict house price using a linear regression model fit on these 10 data points we can use the following equation

$$SalePrice = 63644 + 236 * GarageArea$$

Find regression co-efficients for `GarageArea` using OLS

In this task, you will be calculating the regression coefficients as well as plotting a regression line for `GarageArea` against the target variable `SalePrice`

Instructions

- First define a function `estimate_coef(x,y)` to estimate the coefficients of regression which takes two arguments; a feature (x) and a target variable(y) and returns a tuple containing the two coefficients for regression. Consult the material on the topic if you get stuck at any point
- Now, define a function `plot_regression_line(x,y,b)` which will plot the regression line and takes three arguments; a feature(x), a target variable(y) and a tuple(b) which consists of the regression coefficients
- Now first estimate the coefficients of the regression line for the feature `GarageArea` with respect to the target using the `estimate_coef()` function that you had defined. Store the coefficients in variable `values`
- Using the output of the above step plot the regression line and with the help of `plot_regression_line()` and display it

For `estimate_coef()`

- Calculate mean of variables x and y
- Calculate cross deviations using

$$SS_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

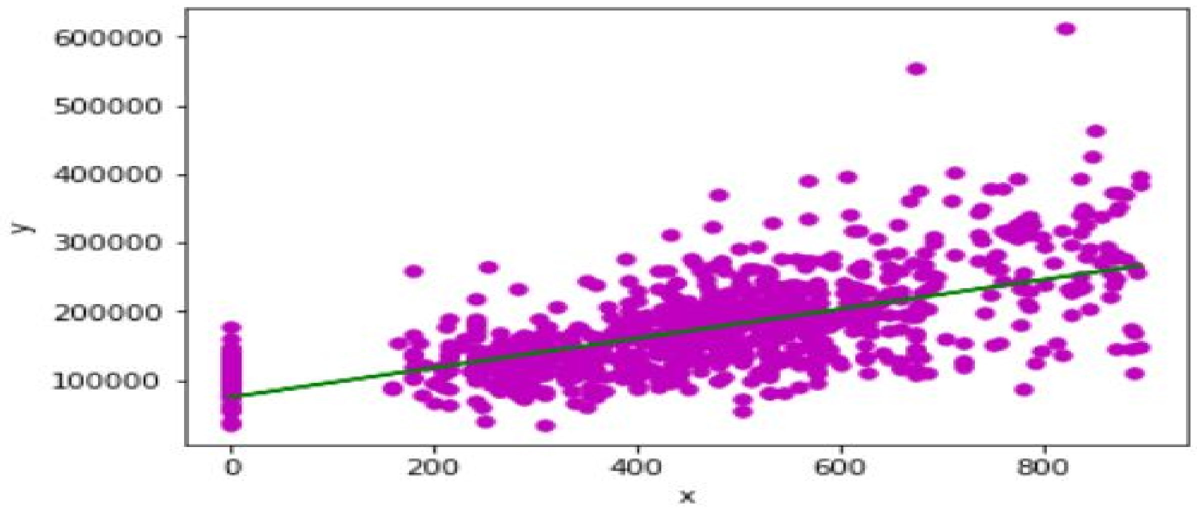
$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

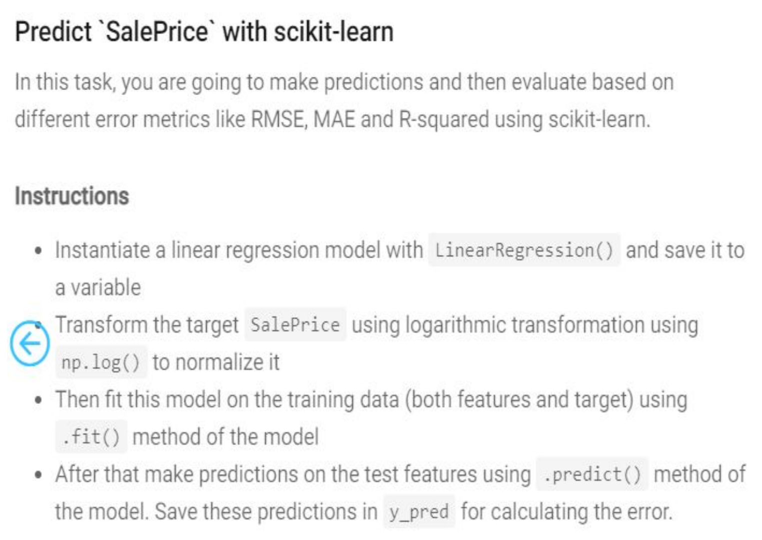
- Calculate values of coefficients using

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

```python
# import packages
import matplotlib.pyplot as plt
# Code starts here
def estimate_coef(x, y):
   # number of observations/points
   n = np.size(x)
   # mean of x and y vector
   m_x, m_y = np.mean(x), np.mean(y)
   # calculating cross-deviation and deviation about x
   SS_xy = np.sum(y*x)- n*m_y*m_x
   SS_xx = np.sum(x*x) - n*m_x*m_x
   # calculating regression coefficients
   b_1 = SS_xy / SS_xx
   b_0 = m_y - b_1*m_x
   return(b_0, b_1)
def plot_regression_line(x, y, b):
   # plotting the actual points as scatter plot
   plt.scatter(x, y, color = "m",
              marker = "o", s = 30)
   # predicted response vector
   y_pred = b[0] + b[1]*x
   # plotting the regression line
   plt.plot(x, y_pred, color = "g")
   # putting labels
   plt.xlabel('x')
   plt.ylabel('y')
   # function to show plot
   plt.show()
# Visualize your results
values = estimate_coef(X_train['GarageArea'], y_train)
plot_regression_line(X_train['GarageArea'], y_train, values)
# Code ends here
```

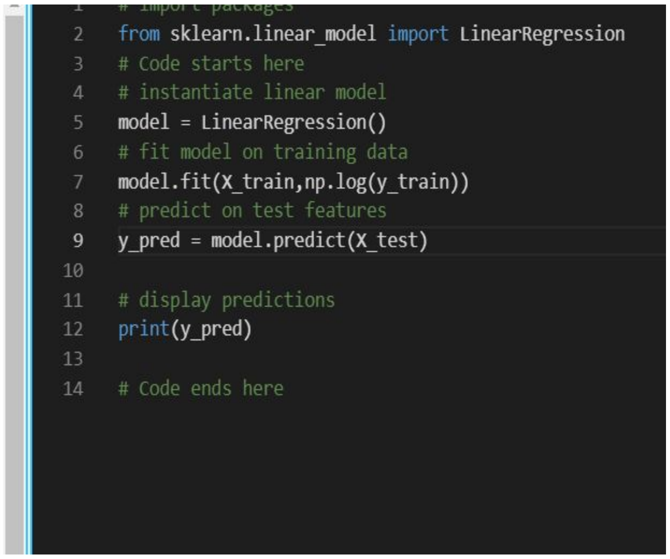



Predict `SalePrice` with scikit-learn

In this task, you are going to make predictions and then evaluate based on different error metrics like RMSE, MAE and R-squared using scikit-learn.

Instructions

- Instantiate a linear regression model with `LinearRegression()` and save it to a variable
- Transform the target `SalePrice` using logarithmic transformation using `np.log()` to normalize it
- Then fit this model on the training data (both features and target) using `.fit()` method of the model
- After that make predictions on the test features using `.predict()` method of the model. Save these predictions in `y_pred` for calculating the error.



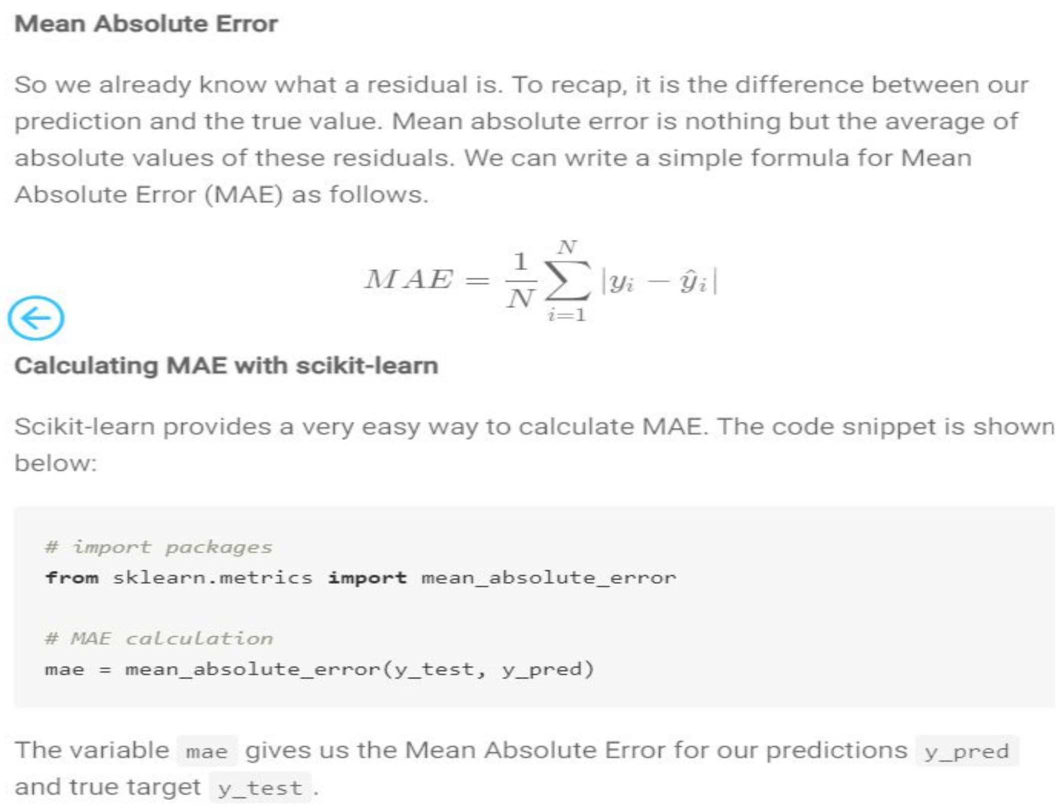
```python
from sklearn.linear_model import LinearRegression
# Code starts here
# instantiate linear model
model = LinearRegression()
# fit model on training data
model.fit(X_train,np.log(y_train))
# predict on test features
y_pred = model.predict(X_test)

# display predictions
print(y_pred)

# Code ends here
```

Evaluation : Mean Absolute Error

Now that you have fitted your model on training data, it is time to test it on unseen data. You also need to have some kind of measure to quantify the performance of the model. This measure is captured by what we call the error metrics and there are many different forms depending on the problem statement.

We have a regression problem at hand and the different types of error metrics that can be used are:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-Squared



Mean Absolute Error

So we already know what a residual is. To recap, it is the difference between our prediction and the true value. Mean absolute error is nothing but the average of absolute values of these residuals. We can write a simple formula for Mean Absolute Error (MAE) as follows.

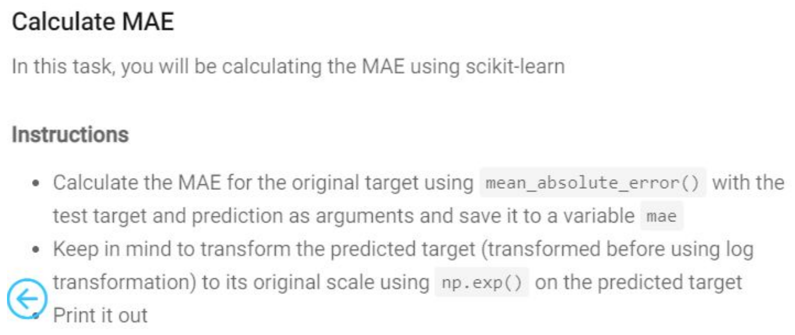
$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Calculating MAE with scikit-learn

Scikit-learn provides a very easy way to calculate MAE. The code snippet is shown below:

```python
# import packages
from sklearn.metrics import mean_absolute_error

# MAE calculation
mae = mean_absolute_error(y_test, y_pred)
```

The variable `mae` gives us the Mean Absolute Error for our predictions `y_pred` and true target `y_test`.

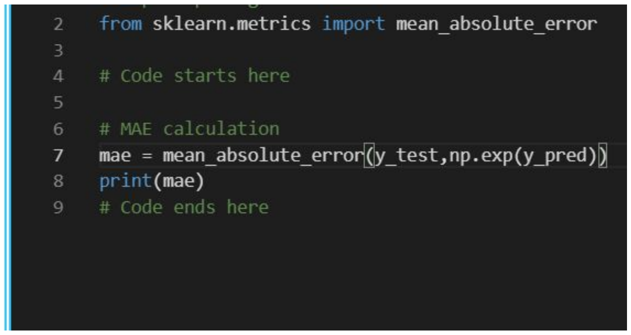


Calculate MAE

In this task, you will be calculating the MAE using scikit-learn

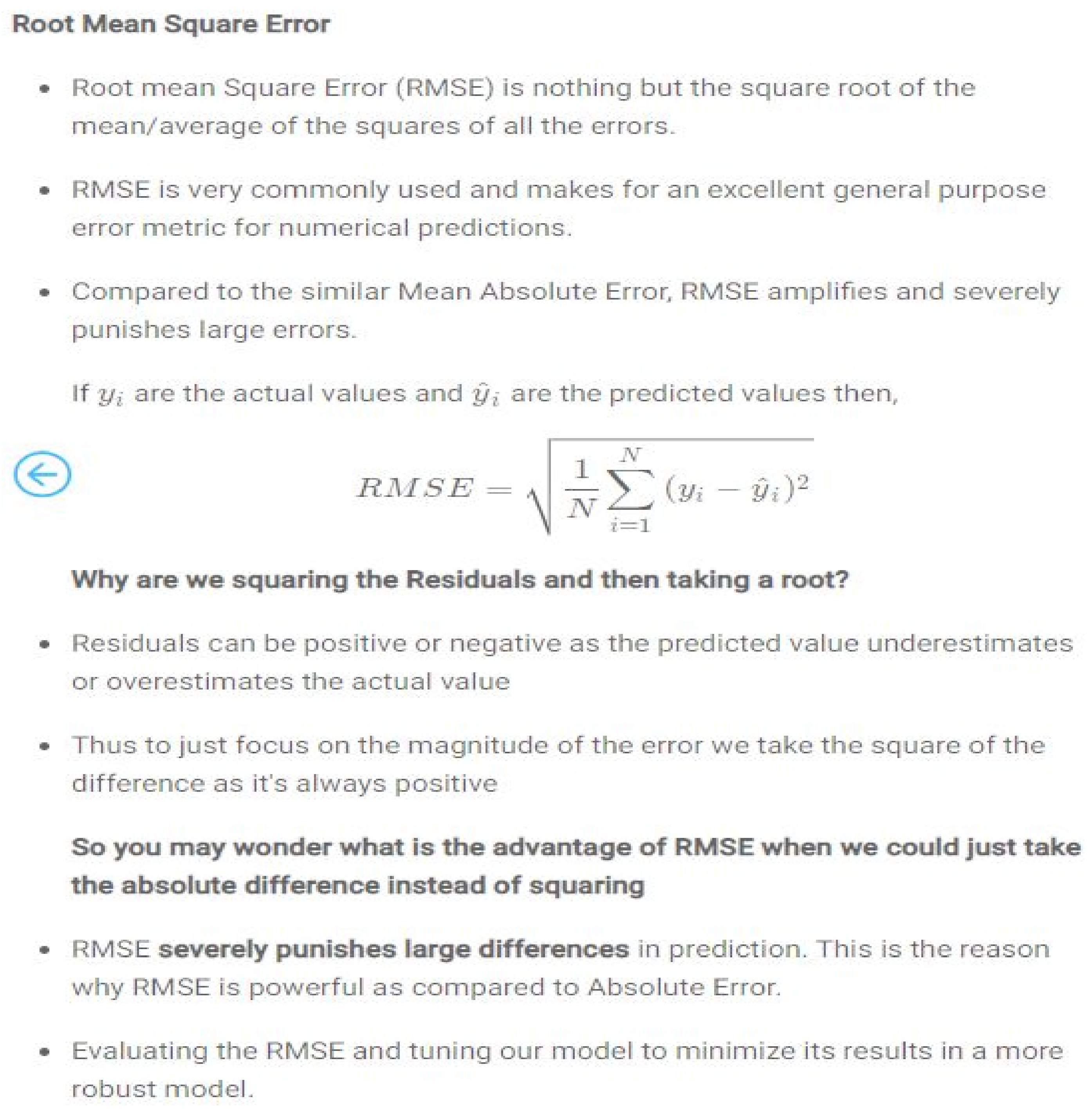
Instructions

- Calculate the MAE for the original target using `mean_absolute_error()` with the test target and prediction as arguments and save it to a variable `mae`
- Keep in mind to transform the predicted target (transformed before using log transformation) to its original scale using `np.exp()` on the predicted target
- Print it out



```python
from sklearn.metrics import mean_absolute_error

# Code starts here

# MAE calculation
mae = mean_absolute_error(y_test,np.exp(y_pred))
print(mae)
# Code ends here
```

Evaluation : Root Mean Squared Error



Root Mean Square Error

- Root mean Square Error (RMSE) is nothing but the square root of the mean/average of the squares of all the errors.
- RMSE is very commonly used and makes for an excellent general purpose error metric for numerical predictions.
- Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.

If y_i are the actual values and $\hat{y}_i$ are the predicted values then,

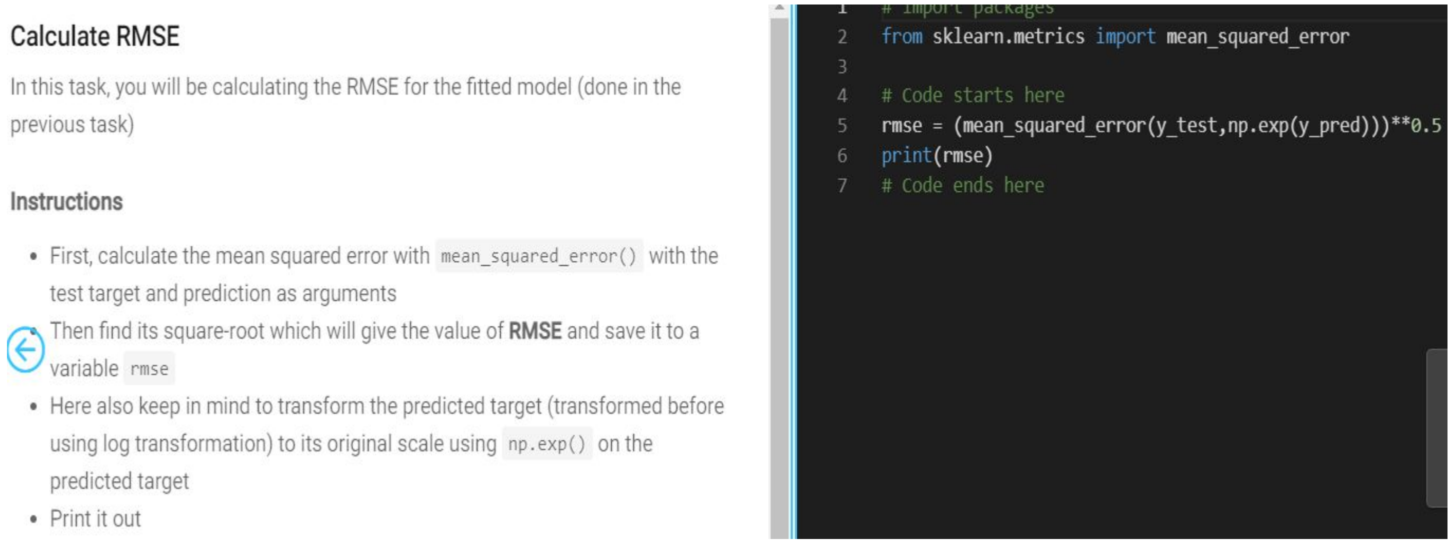
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Why are we squaring the Residuals and then taking a root?

- Residuals can be positive or negative as the predicted value underestimates or overestimates the actual value
- Thus to just focus on the magnitude of the error we take the square of the difference as it's always positive

So you may wonder what is the advantage of RMSE when we could just take the absolute difference instead of squaring

- RMSE **severely punishes large differences** in prediction. This is the reason why RMSE is powerful as compared to Absolute Error.
- Evaluating the RMSE and tuning our model to minimize its results in a more robust model.


Calculate RMSE

In this task, you will be calculating the RMSE for the fitted model (done in the previous task)

Instructions

- First, calculate the mean squared error with `mean_squared_error()` with the test target and prediction as arguments
- Then find its square-root which will give the value of **RMSE** and save it to a variable `rmse`
- Here also keep in mind to transform the predicted target (transformed before using log transformation) to its original scale using `np.exp()` on the predicted target
- Print it out

```
# import packages
from sklearn.metrics import mean_squared_error

# Code starts here
rmse = (mean_squared_error(y_test,np.exp(y_pred)))**0.5
print(rmse)
# Code ends here
```

Evaluation : R square

What is R-Squared?

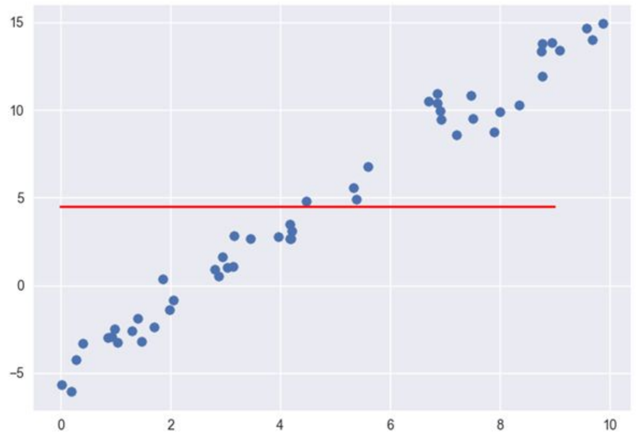
R-squared is a statistical measure of how close the data are to the fitted regression line i.e. it measures the goodness of fit of a straight line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

It is a measure of the proportion of variability in the response that is explained by the regression model.

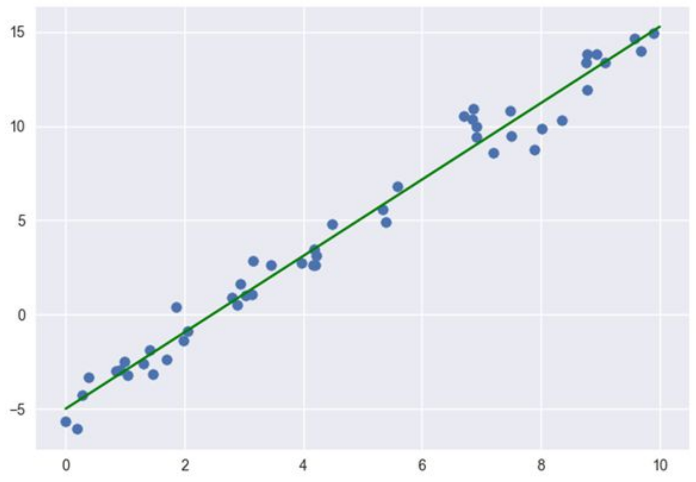
Mathematically,
$$R^2 = \frac{\text{Explained variation}}{\text{Total variation}}$$

R-squared is always between 0 and 100%:

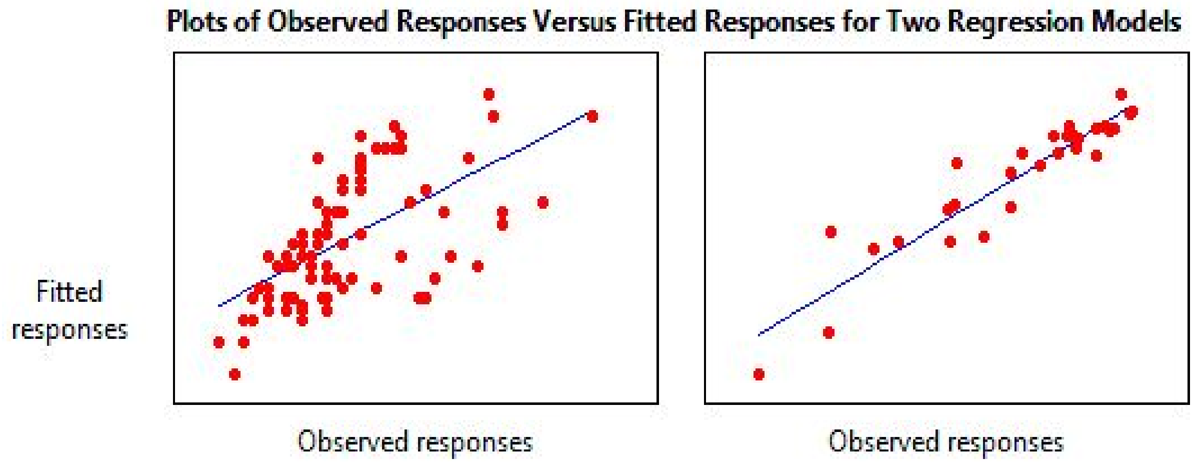
1. 0 % indicates that the model explains none of the variability of the response data around its mean.



2. 100% indicates that the model explains all the variability of the response data around its mean.



Graphical Intuition behind $R2$



In the above figure, you are given two plots for two different regression models which have fitted responses (predicted) on the Y-axis and observed responses (true) on the x-axis. In linear regression, you want the predicted values to be close to the actual values. So to have a good fit, that plot should resemble a straight line at 45 degrees.

The regression model on the left accounts for around 30.0% of the variance while the one on the right accounts for around 90%. So, the more the variance that is accounted for by the regression model, the closer the data points will fall to the fitted regression line.

Theoretically, if a model could explain 100% of the variance, the fitted values would always equal the observed values and, therefore, all the data points would fall on the fitted regression line.

Mathematical Intuition behind R^2

Before we get into calculating R^2, let's understand what the different Sum of Squares for our model are:

- In RMSE, we have already been introduced to Squared Residuals which is also called Error Sum of Squares (SSE)

$$SSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- Additionally, we have the Total Sum of Squares (SST) which is nothing but the Squared difference between the Actual Values (y_i) and the Mean of our dataset ($\overline{y}_i$). This is also the baseline model where we predict all the values as the mean of the true values. It doesn't make use of any independent variables to predict the value of dependent variable Y. Instead, it uses the mean of the observed responses of dependent variable Y and always predicts this mean as the value of Y.

  R-squared simply explains how good is your model when compared to this baseline model.

$$SS(Total) = \sum_{i=1}^{N} (y_i - \overline{y}_i)^2$$

- Additionally, we have the Total Sum of Squares (SST) which is nothing but the Squared difference between the Actual Values (y_i) and the Mean of our dataset ($\overline{y}_i$). This is also the baseline model where we predict all the values as the mean of the true values. It doesn't make use of any independent variables to predict the value of dependent variable Y. Instead, it uses the mean of the observed responses of dependent variable Y and always predicts this mean as the value of Y.

  R-squared simply explains how good is your model when compared to this baseline model.

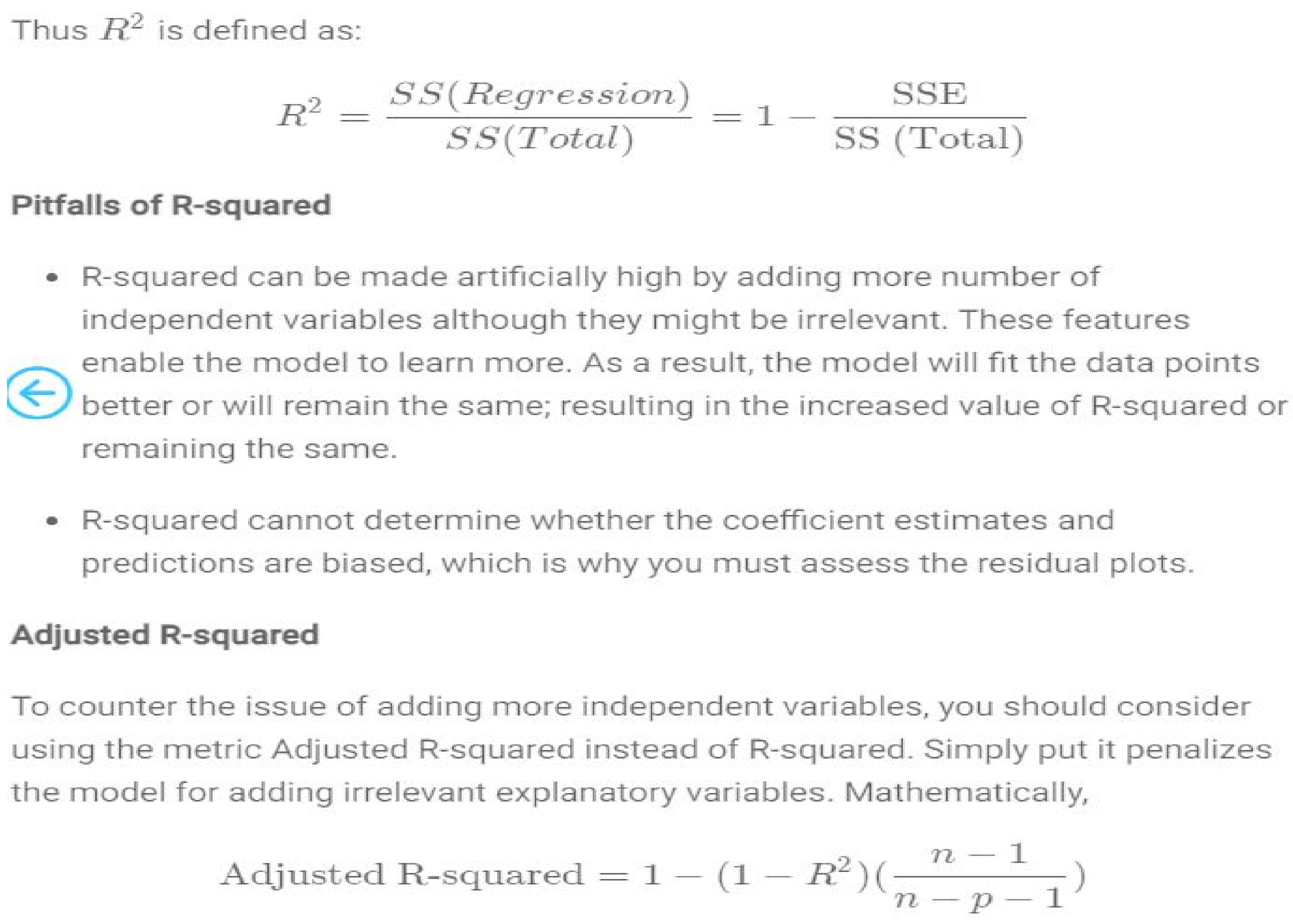
$$SS(Total) = \sum_{i=1}^{N} (y_i - \overline{y}_i)^2$$

- And we also have Regression Sum of Squares, which is the squared difference between the Predicted values ($\hat{y}_i$) and the Mean ($\overline{y}_i$)

$$SS(Regression) = \sum_{i=1}^{N} (\hat{y}_i - \overline{y}_i)^2$$

Now, intuitively, we can see that:

$$SS(Total) = SS(Regression) + SSE$$



Thus R^2 is defined as:

$$R^2 = \frac{SS(Regression)}{SS(Total)} = 1 - \frac{\text{SSE}}{\text{SS (Total)}}$$

Pitfalls of R-squared

- R-squared can be made artificially high by adding more number of independent variables although they might be irrelevant. These features enable the model to learn more. As a result, the model will fit the data points better or will remain the same; resulting in the increased value of R-squared or remaining the same.
- R-squared cannot determine whether the coefficient estimates and predictions are biased, which is why you must assess the residual plots.

Adjusted R-squared

To counter the issue of adding more independent variables, you should consider using the metric Adjusted R-squared instead of R-squared. Simply put it penalizes the model for adding irrelevant explanatory variables. Mathematically,

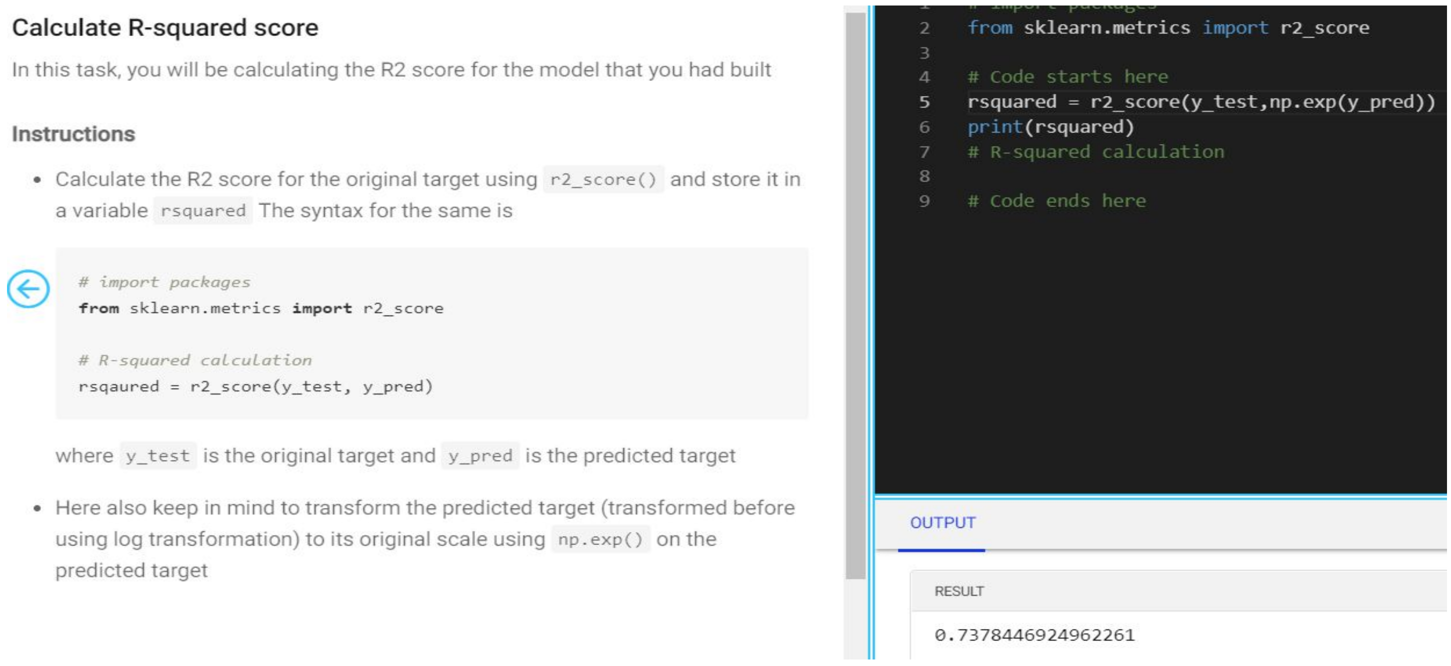
$$\text{Adjusted R-squared} = 1 - (1 - R^2)(\frac{n-1}{n-p-1})$$

where

n = Number of data points

p = Number of explanatory/independent variables

So how is R-squared different from Adjusted R-squared? R-squared tells you how well your model fits the data points whereas Adjusted R-squared tells you how important is a particular feature to your model.



Calculate R-squared score

In this task, you will be calculating the R2 score for the model that you had built

Instructions

- Calculate the R2 score for the original target using `r2_score()` and store it in a variable `rsquared` The syntax for the same is

```
# import packages
from sklearn.metrics import r2_score

# R-squared calculation
rsqaured = r2_score(y_test, y_pred)
```

  where `y_test` is the original target and `y_pred` is the predicted target

- Here also keep in mind to transform the predicted target (transformed before using log transformation) to its original scale using `np.exp()` on the predicted target

```
from sklearn.metrics import r2_score

# Code starts here
rsquared = r2_score(y_test,np.exp(y_pred))
print(rsquared)
# R-squared calculation

# Code ends here
```

OUTPUT

RESULT

```
0.7378446924962261
```

RMSE punishes errors in prediction more severely as compared to MAE.

Ans: True

Explanation:
In RMSE the ‘squared’ nature of this metric helps to deliver more robust results which prevents cancelling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of error term.

Real Life Use Case

Business Objective

A home is often the major and most important investment a person makes in his or her lifetime. The prices of land and real estate have increased exponentially in the last two decades and have caused overpricing of commercial or residential property.

Our client, XYZ Company, is one of the leading property dealers in the country. It runs an online website for homeowners, buyers, sellers, etc. It provides real estate data to the customer at no extra cost. The major source of income for our company is through advertisement. As per ABC data, more than 70% of homebuyers now use the internet as an integral part of their home buying experience. The company wants to ensure that its customers have a trusted way to monitor the asset they would be investing in.

To achieve this, the company wants to analyze the past data of the housing market and provide an estimate of the property value to the customer at no cost.

Solution Methodology

The property value was estimated through Linear Regression. Here, the target variable - property value, is a quantitative value, which can be represented as a function of other features like, property*area, year*of*construction, location, number*of_rooms, etc. Around 50 features were identified to contribute significantly to the property value.

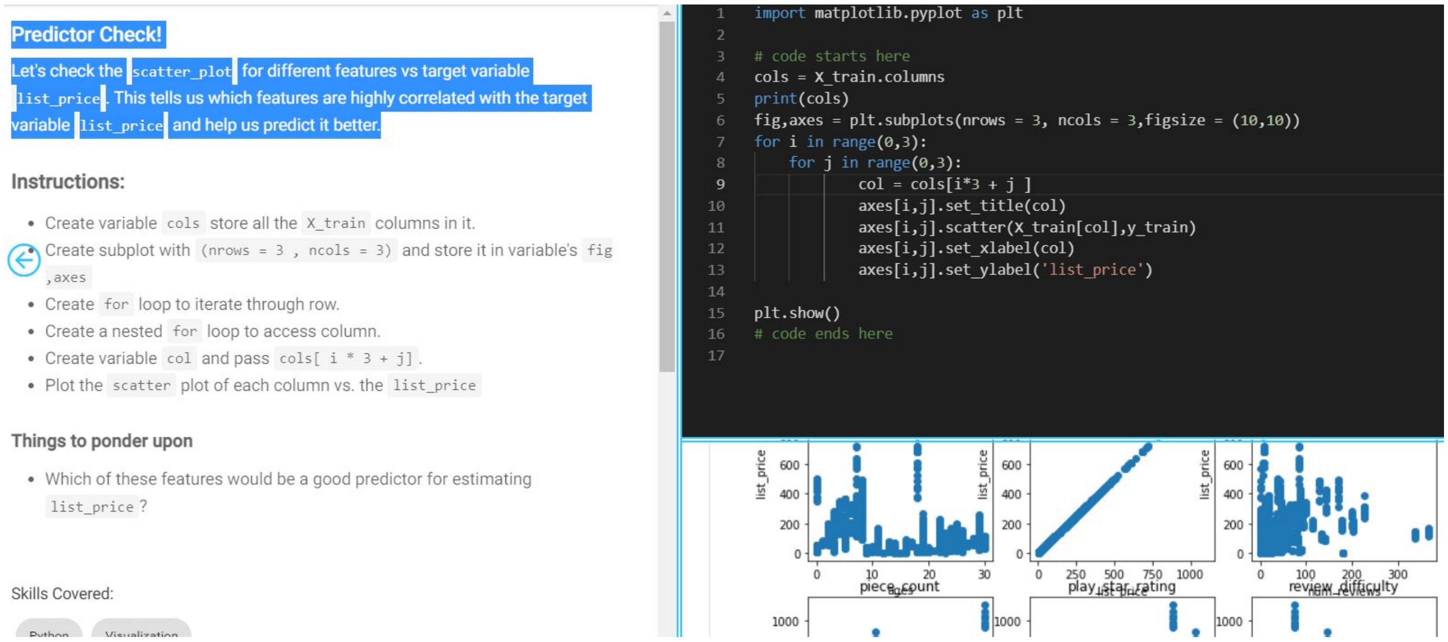
The data was prepared and trained on Linear Regression to predict the property value.

Business Impact

The trained model was able to make

- Better predictions for newer homes (built between 1950-2000).
- Better predictions for bigger homes.

This gave a better experience to the customer and resulted in a 200% increase in sales of houses in the last quarter of 2018, as compared to the previous year.



Predictor Check!

Let's check the `scatter_plot` for different features vs target variable `list_price`. This tells us which features are highly correlated with the target variable `list_price` and help us predict it better.

Instructions:

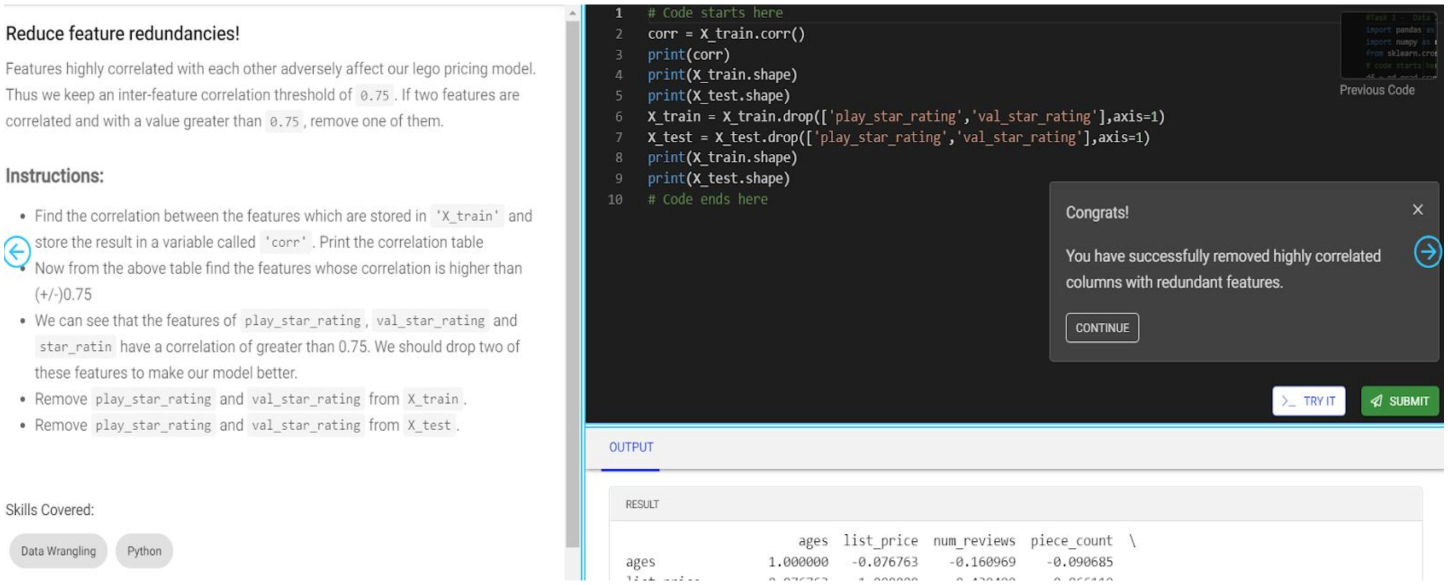
- Create variable `cols` store all the `X_train` columns in it.
- Create subplot with `(nrows = 3 , ncols = 3)` and store it in variable's `fig ,axes`
- Create `for` loop to iterate through row.
- Create a nested `for` loop to access column.
- Create variable `col` and pass `cols[ i * 3 + j]`.
- Plot the `scatter` plot of each column vs. the `list_price`

Things to ponder upon

- Which of these features would be a good predictor for estimating `list_price`?

Skills Covered:

```
import matplotlib.pyplot as plt

# code starts here
cols = X_train.columns
print(cols)
fig,axes = plt.subplots(nrows = 3, ncols = 3,figsize = (10,10))
for i in range(0,3):
    for j in range(0,3):
            col = cols[i*3 + j ]
            axes[i,j].set_title(col)
            axes[i,j].scatter(X_train[col],y_train)
            axes[i,j].set_xlabel(col)
            axes[i,j].set_ylabel('list_price')

plt.show()
# code ends here
```



Reduce feature redundancies!

Features highly correlated with each other adversely affect our lego pricing model. Thus we keep an inter-feature correlation threshold of `0.75`. If two features are correlated and with a value greater than `0.75`, remove one of them.

Instructions:

- Find the correlation between the features which are stored in `'X_train'` and store the result in a variable called `'corr'`. Print the correlation table
- Now from the above table find the features whose correlation is higher than (+/-)0.75
- We can see that the features of `play_star_rating`, `val_star_rating` and `star_ratin` have a correlation of greater than 0.75. We should drop two of these features to make our model better.
- Remove `play_star_rating` and `val_star_rating` from `X_train`.
- Remove `play_star_rating` and `val_star_rating` from `X_test`.

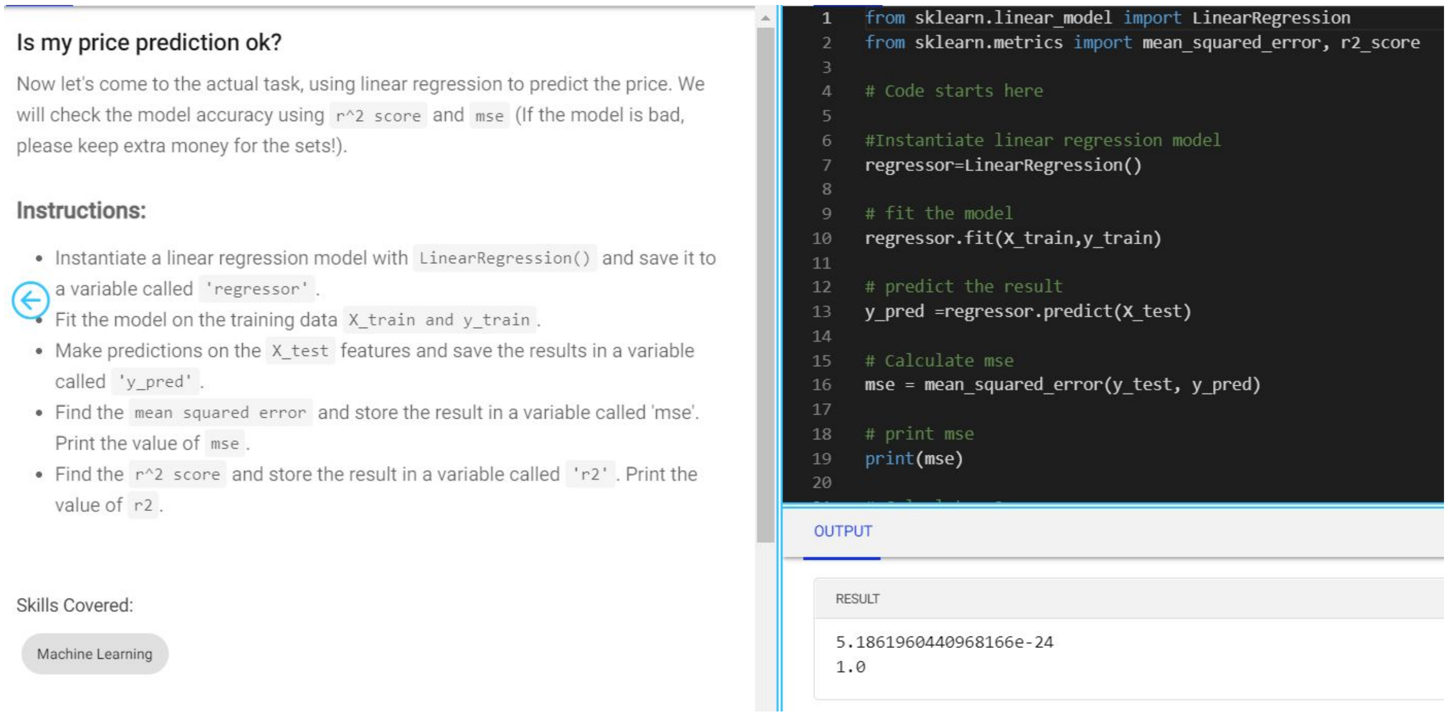
Skills Covered:

```
# Code starts here
corr = X_train.corr()
print(corr)
print(X_train.shape)
print(X_test.shape)
X_train = X_train.drop(['play_star_rating','val_star_rating'],axis=1)
X_test = X_test.drop(['play_star_rating','val_star_rating'],axis=1)
print(X_train.shape)
print(X_test.shape)
# Code ends here
```

Congrats!

You have successfully removed highly correlated columns with redundant features.

```
              ages  list_price  num_reviews  piece_count  \
ages      1.000000   -0.076763    -0.160969    -0.090685
```



Is my price prediction ok?

Now let's come to the actual task, using linear regression to predict the price. We will check the model accuracy using `r^2 score` and `mse` (If the model is bad, please keep extra money for the sets!).

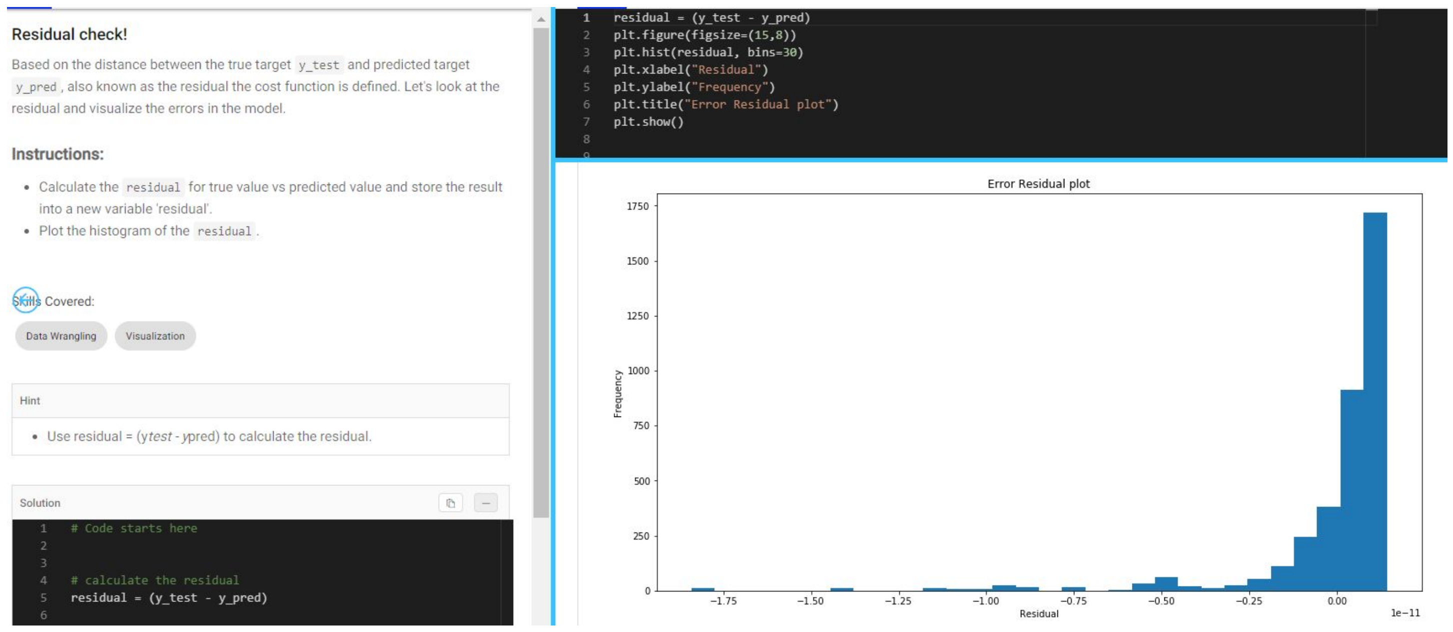
Instructions:

- Instantiate a linear regression model with `LinearRegression()` and save it to a variable called `'regressor'`.
- Fit the model on the training data `X_train and y_train`.
- Make predictions on the `X_test` features and save the results in a variable called `'y_pred'`.
- Find the `mean squared error` and store the result in a variable called 'mse'. Print the value of `mse`.
- Find the `r^2 score` and store the result in a variable called `'r2'`. Print the value of `r2`.

Skills Covered:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Code starts here

#Instantiate linear regression model
regressor=LinearRegression()

# fit the model
regressor.fit(X_train,y_train)

# predict the result
y_pred =regressor.predict(X_test)

# Calculate mse
mse = mean_squared_error(y_test, y_pred)

# print mse
print(mse)
```

```
5.1861960440968166e-24
1.0
```


Residual check!

Based on the distance between the true target `y_test` and predicted target `y_pred`, also known as the residual the cost function is defined. Let's look at the residual and visualize the errors in the model.

Instructions:

- Calculate the `residual` for true value vs predicted value and store the result into a new variable 'residual'.
- Plot the histogram of the `residual`.

Skills Covered:

Data Wrangling   Visualization

Hint

- Use residual = (*ytest* - *y*pred) to calculate the residual.

Solution

```
# Code starts here


# calculate the residual
residual = (y_test - y_pred)
```

```
residual = (y_test - y_pred)
plt.figure(figsize=(15,8))
plt.hist(residual, bins=30)
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.title("Error Residual plot")
plt.show()
```

Questions:

1. Changing the units of measurement of the Y variable will affect all but which one of the following?

Ans: R squared for the regression

Explanation:

R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale. If you change the units of measurement R square in regression gets affected.

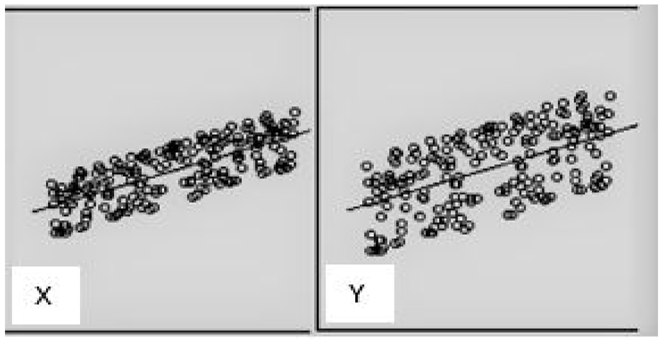
2.A “Linear regression” model perfectly fits the training data (train error is zero). Which of the following statement is true?

Ans: Neither of the above

Explanation:

Test error may be zero if there no noise in test data. In other words, it will be zero, if the test data is perfect representative of train data but not always.

3.For the following image which one of the statements is true?



Ans: Both have the same sum of residuals

Explanation: sum of residuals are always zero.

4.The regression line is drawn such that:
Ans: The sum of the absolute errors is as small as possible.

Explanation:
Regression line draws in such a way that we will get the sum of the absolute errors is as small as possible.

5.Which of these problems would be more suitable to apply linear regression?
Ans: Find the price of car

Explanation:
In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values. Find the price of car is giving continuous values.

6.In a simple linear regression model the slope coefficient measures
Ans: Indicates by how many percent Y increases, given a one percent increase in X.

Explanation:
In a simple linear regression model the slope coefficient measures indicates by how many percent Y increases, given a one percent increase in X.

7.What will happen to the value of R-squared if we increase the number of features?
Ans: Increase or Remain the same

Explanation:
As we increase the features more probabily r square will increase of remains the same.

8.Which of the following metrics can be used for evaluating linear regression models?
Ans: R Squared, Adjusted R Squared, MSE/ MAE

Explanation:
Apart from the accuracy all of them are use for evaluating the linear regression model

9.Which of the following is NOT a possible value of the correlation coefficient?
Ans: 1.2

Explanation:
Correlation coefficient lies between -1 to 1.

10.What is the slope of a line parallel to the X-axis?

Ans: 0

Explanation:
The x-axis is a horizontal line with the equation $y=0$. There are an infinite number of lines that are parallel to the x-axis, y=0 .

Examples: $y=4$, $y=-2$, $y=9.5$

All horizontal lines have slope of 0.

If lines are parallel then they have the same slope.

The slope of a line parallel to the x -axis is 0.