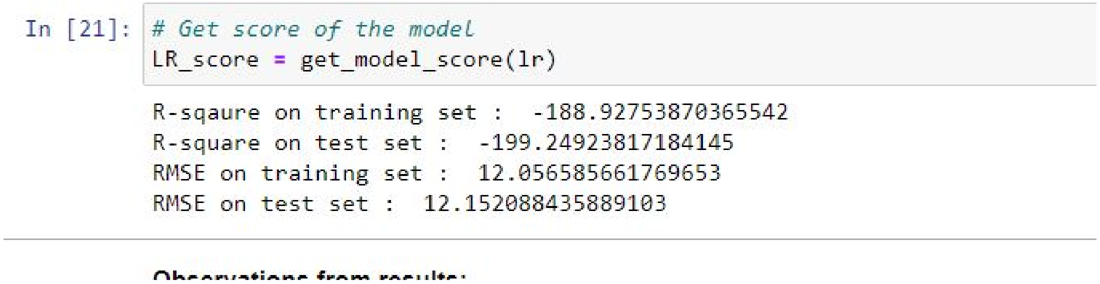
FAQs - Used Cars Price Prediction

1. Why I'm receiving a negative R-square score during Linear Regression model evaluation through the get_model_score() method?



```
In [21]: # Get score of the model
         LR_score = get_model_score(lr)

         R-sqaure on training set :  -188.92753870365542
         R-square on test set :  -199.24923817184145
         RMSE on training set :  12.056585661769653
         RMSE on test set :  12.152088435889103
```

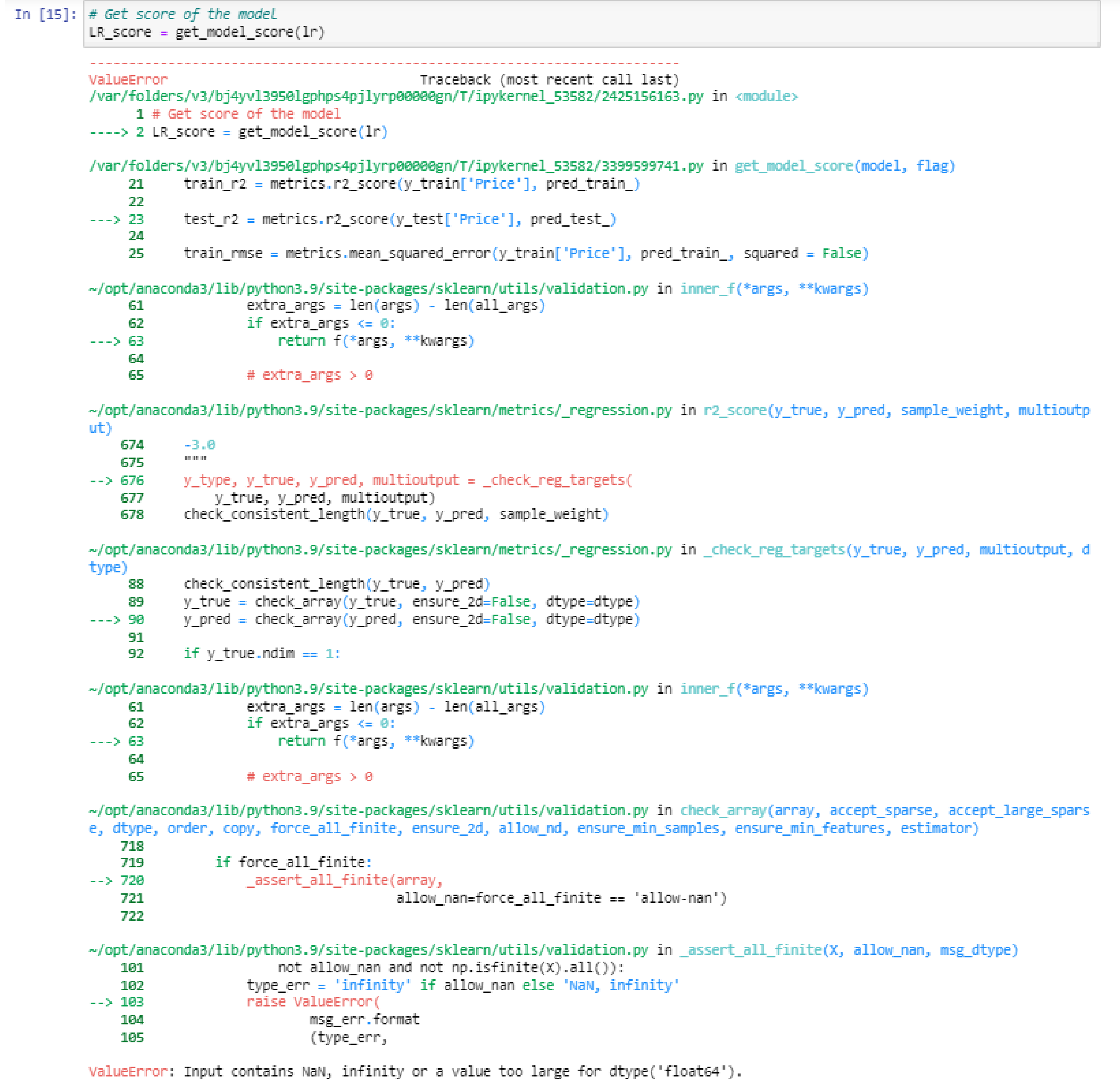
Observations from results:

You are getting a negative score on both the train and test data because the results were comparing it with the **'price_log'** variable instead of the 'Price' variable in the get_model_score() function. We need to compare it with only the Price column because we are applying the exponential transformation to the predictions/results from the model which can restore the results from logarithm form to normal form. So, please compare the model predictions with **the Price** variable only.

note: By default, we had given the Price column only in the get_model_score() function, so please do not change it to the price_log variable.

2. I'm receiving the following error in my linear regression model even though my dataset has no null values.

ValueError: Input contains NaN, infinity or a value too large for dtype('float64').



```
In [15]: # Get score of the model
         LR_score = get_model_score(lr)

         ---------------------------------------------------------------------------
         ValueError                                Traceback (most recent call last)
         /var/folders/v3/bj4yvl3950lgphps4pjlyrp00000gn/T/ipykernel_53582/2425156163.py in <module>
               1 # Get score of the model
         ----> 2 LR_score = get_model_score(lr)

         /var/folders/v3/bj4yvl3950lgphps4pjlyrp00000gn/T/ipykernel_53582/3399599741.py in get_model_score(model, flag)
              21     train_r2 = metrics.r2_score(y_train['Price'], pred_train_)
              22 
         ---> 23     test_r2 = metrics.r2_score(y_test['Price'], pred_test_)
              24 
              25     train_rmse = metrics.mean_squared_error(y_train['Price'], pred_train_, squared = False)

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in inner_f(*args, **kwargs)
              61             extra_args = len(args) - len(all_args)
              62             if extra_args <= 0:
         ---> 63                 return f(*args, **kwargs)
              64 
              65             # extra_args > 0

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_regression.py in r2_score(y_true, y_pred, sample_weight, multioutp
         ut)
             674     -3.0
             675     """
         --> 676     y_type, y_true, y_pred, multioutput = _check_reg_targets(
             677         y_true, y_pred, multioutput)
             678     check_consistent_length(y_true, y_pred, sample_weight)

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_regression.py in _check_reg_targets(y_true, y_pred, multioutput, d
         type)
              88     check_consistent_length(y_true, y_pred)
              89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
         ---> 90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)
              91 
              92     if y_true.ndim == 1:

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in inner_f(*args, **kwargs)
              61             extra_args = len(args) - len(all_args)
              62             if extra_args <= 0:
         ---> 63                 return f(*args, **kwargs)
              64 
              65             # extra_args > 0

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_spars
         e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
             718 
             719         if force_all_finite:
         --> 720             _assert_all_finite(array,
             721                                allow_nan=force_all_finite == 'allow-nan')
             722 

         ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype)
             101                 not allow_nan and not np.isfinite(X).all()):
             102             type_err = 'infinity' if allow_nan else 'NaN, infinity'
         --> 103             raise ValueError(
             104                     msg_err.format
             105                     (type_err,

         ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

The error that you are getting usually happens when there are missing values or infinity values present in the data. To cross verify, whether the null values are there present in the data, please use the below line

data.isna().sum()

If the above code returns zeros to every feature then there are no null values in the data. If the null values are present in the data, please either impute with the appropriate technique or drop the null values.

If there are no null values present in the data, then the error might occur due to the presence of infinity values for the linear regression model predictions, which would result in an error during the model r2_score calculation. There are some possible reasons for this, please have a look at the below checklist:

1. It depends on how you are imputing the missing values in the data. Please impute the missing values as per the type and nature of that specific column.

For example, if the column was numerical, and it is not skewed then impute it with the mean of its distribution of the data. Also, when imputing the missing values through the aggregation of **Brand** and **model** columns leads to the distribution of data that is giving the infinity values during the model prediction. So we would recommend imputing only the **Seats** column with the aggregation of Brand and model columns rest will be imputed directly with mean or median.

2. It happens when we use scaled data for implementing the model. So, avoid the training data that was scaled, especially the **new_price** column in order to get rid of the infinity values.

3. It happens if you are including both the normal and log-transformed values in the training data. So, please be cautious when creating the dependent and independent variables. To get to know what features are included in the independent feature, you can take the help of the below code:

```
X.columns
```

Observe the output and remove the columns where the column and its log-transformed columns are present in the data. For example, Kilometers_Driven and Kilometers_Driven_log both the columns are available in X, so please drop either Kilometer_Driven or Kilometers_Driven_log.

4. It happens when we include more features, i.e., please be cautious when creating the Brand and Model columns from the Name column. If the features are high around 2000+ features can lead to infinity values during the model predictions. So, please keep the number of features less than 2000. Below are the possible reasons and solutions for the creation of 2000+ columns.

a. When creating the Brand and Model columns from the name column in the data, please first convert every name to lowercase in order to avoid creating 2 different features for the same car name.

b. Sometimes **mileage** column should get converted to an object data type and every value gets treated as a category which leads to more features. So, please keep an eye on the mileage column, and make sure that the column and its values should be numerical in nature.

3. What is meant by the Final solution notebook? How to submit the final notebook?

The final solution notebook means the improvement and extension of Milestone submission that includes starting from the data exploration to ending with possible further recommendations.

4. I'm getting a very low score for the base model in comparison to the tuned tree-based models. Why is it happening?

You might not be using the proper scoring metric for the Decision Tree Regressor and Random Forest Regressor. For regression, the scoring metric should be r2_score or some other regression metric, not **recall_score,** which is used for a classification problem. We have provided the code to tune the random forest regressor in the notebook, which can be applied to the decision tree regressor as well.

5. Getting the below error while invoking the get_model_score() function.

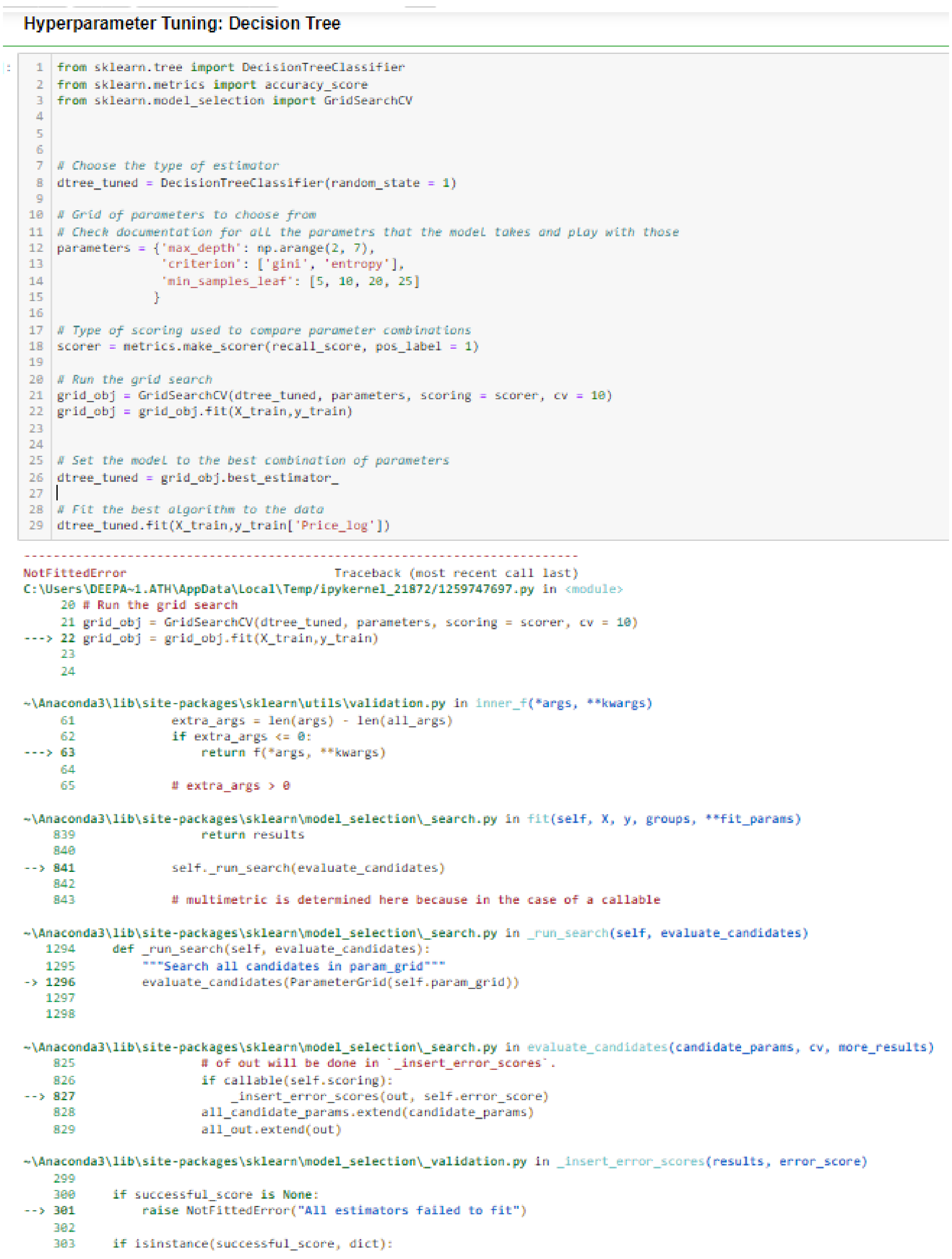
#error #metrics #libraries #capstone

```
NameError: name 'metrics' is not defined
```

This error might be because you have not imported one of the libraries. To use the metrics in the sklearn library, we need to import them into the notebook first. Please run the below code to import it into your working environment. After importing it into the working environment, everything executes smoothly.

```
from sklearn import metrics
```

6. Getting the below error while tuning the decision tree model. How do resolve the error?



Hyperparameter Tuning: Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV



# Choose the type of estimator
dtree_tuned = DecisionTreeClassifier(random_state = 1)

# Grid of parameters to choose from
# Check documentation for all the parametrs that the model takes and play with those
parameters = {'max_depth': np.arange(2, 7),
              'criterion': ['gini', 'entropy'],
              'min_samples_leaf': [5, 10, 20, 25]
             }

# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(recall_score, pos_label = 1)

# Run the grid search
grid_obj = GridSearchCV(dtree_tuned, parameters, scoring = scorer, cv = 10)
grid_obj = grid_obj.fit(X_train,y_train)


# Set the model to the best combination of parameters
dtree_tuned = grid_obj.best_estimator_

# Fit the best algorithm to the data
dtree_tuned.fit(X_train,y_train['Price_log'])
```

```
---------------------------------------------------------------------------
NotFittedError                            Traceback (most recent call last)
C:\Users\DEEPA~1.ATH\AppData\Local\Temp/ipykernel_21872/1259747697.py in <module>
     20 # Run the grid search
     21 grid_obj = GridSearchCV(dtree_tuned, parameters, scoring = scorer, cv = 10)
---> 22 grid_obj = grid_obj.fit(X_train,y_train)
     23
     24

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in fit(self, X, y, groups, **fit_params)
    839                 return results
    840
--> 841             self._run_search(evaluate_candidates)
    842
    843             # multimetric is determined here because in the case of a callable

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in _run_search(self, evaluate_candidates)
   1294     def _run_search(self, evaluate_candidates):
   1295         """Search all candidates in param_grid"""
-> 1296         evaluate_candidates(ParameterGrid(self.param_grid))
   1297
   1298

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in evaluate_candidates(candidate_params, cv, more_results)
    825                 # of out will be done in `_insert_error_scores`.
    826                 if callable(self.scoring):
--> 827                     _insert_error_scores(out, self.error_score)
    828                 all_candidate_params.extend(candidate_params)
    829                 all_out.extend(out)

~\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _insert_error_scores(results, error_score)
    299
    300     if successful_score is None:
--> 301         raise NotFittedError("All estimators failed to fit")
    302
    303     if isinstance(successful_score, dict):
```

The error you are getting is because you are implementing a classification algorithm with classification parameters to solve a regression problem. Here, the dependent variable is continuous, which cannot be used to implement classification algorithms on it. To avoid the error, please use DecisionTreeRegressor along with its appropriate parameters related to regression.

7. Getting the below error while splitting the data into training and testing. How to resolve this error?

too many values to unpack (expected 3)

The error you are getting is due to the lack of values to unpack while splitting the data. To avoid the error, you must pass enough variables to store both the training and the testing data. To do so, please follow the below lines of code:

```
# Splitting data into training and test set:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
print(X_train.shape, X_test.shape)
```

8. When you divide the dataset into X and y, why are you keeping all the variables? Why are you not getting rid of those variables that were already transformed (like Engine, Power, Price)? what was the point in doing the transformation otherwise? Also, you cannot run a regression with a variable and its transformation, can you?

Yes, you are correct. Please drop either log_transformed variables or normal variables based on the algorithms that you want to use in this use case.

9. When it comes to the data, I do not know what exactly "Type of Ownership" is. I do not understand what it means by "First", "Second", etc. I thought these were all second-hand cars. So what's "first"?

Type of Ownership is a feature that explains the ownership of the vehicles. Here, the primary means that the car had one owner previously. Similarly, secondary means the car had two owners previously and so on.

10. In Problem Definition, what exactly are "key questions", and "problem formulation"? What is the difference between the two?

Key Questions define what are the important questions which need to answered using this solution based on the problem statement.
For example: Which factors would affect the price of used cars?

The problem formulation defines how we are going to solve the problem based on the problem statement and key questions.
For example, we have a regression problem at hand where we will try to predict the price of used cars based on several factors such as - Year of manufacturing, Number of seats, Mileage of the car, etc.

11. In "Data Exploration", I did univariate and bivariate analyses. Is that enough?

Yes, it is enough. Data Exploration involves the Univariate Analysis and Bivariate Analysis in the code.

For the last section Proposed Approach, you don't need to code at all, just explain all your observations and insights in a structured way.

12. It would be helpful to know what do you mean by "Proposed approach", which reads as "potential techniques, overall solution design, and measure of success".

Potential Techniques define what were the possible ways to develop the model to solve the problem. It can be suggesting possible algorithms, different pre-processing techniques, etc.

Overall solution design explains what approach you are going to implement to solve the problem. Please explain your entire work pipeline that would start from the data cleaning to providing final recommendations.

The measure of success defines how you are going to validate the model results on the data like specifying the suitable metrics that are used to validate and compare various models.

13. When evaluating the ridge regression model, I am receiving the below error. How to resolve this error?

```
y_true and y_pred have different number of output (1!=2)
```

The error that you are getting is an output mismatch error. While implementing the ridge regression model, you have trained the data with 2 target features, which is leading to the error as every other algorithm is trained with only 1 target feature. To avoid this error, please run the ridge regression as shown below:

```
ridge.fit(X_train, y_train['price_log'])
```

Note: Please fit all the models with only 1 target feature, i.e., **price_log** during the model implementations.

14. I'm getting the below error while implementing the Decision Tree model. How to resolve the error?

```
ValueError: X has 61 features, but DecisionTreeRegressor is expecting 60 features as input
```

The error that you are getting is due to the fact that your model has been trained on 60 features. Hence, the model expects 60 features as the input while giving predictions. You are passing 61 features as input to the trained model, which is resulting in this ValueError. To avoid this error, please pass only 60 input features to the trained model to make the predictions.

15. In the dataset, Price is my target variable. Is it ok to drop if there are null values in Price variable?

Yes, you can drop the null values in the Price column since it is a target variable. It is not a good practice to impute the dependent variable, which might include bias in the data.

16. How to merge two notebooks in Jupyter Notebook?