Random Forest & Decision Tree - Home Exercise:

The **CarPricePrediction.csv** is a dataset that includes various features about cars and the goal is to predict their prices.

Below is an explanation of typical columns you might find in this dataset:

- **symboling** → Risk factor (+3 indicates risky, -3 indicates safe)
- **normalized-losses** → Normalized losses in use as compared to other cars
- **make** → Manufacturer name (categorical)
- **fuel-type** → Type of fuel (categorical)
- **aspiration** → Type of aspiration (categorical)
- **num-of-doors** → Number of doors (categorical)
- **body-style** → Body style of the car (categorical)
- **drive-wheels** → Type of drive wheels (categorical)
- **engine-location** → Location of the engine (categorical)
- **wheel-base** → The wheelbase of the car
- **length** → The length of the car
- **width** → The width of the car
- **height** → The height of the car
- **curb-weight** → The weight of the car without passengers or cargo
- **engine-type** → Type of engine (categorical)
- **num-of-cylinders** → Number of cylinders in the engine (categorical)
- **engine-size** → The size of the engine
- **fuel-system** → Type of fuel system (categorical)
- **bore** → The bore dimension of the engine cylinders
- **stroke** → The stroke dimension of the engine cylinders
- **compression-ratio** → The ratio of cylinder volume
- **horsepower** → The horsepower of the car
- **peak-rpm** → The peak RPM of the engine
- **city-mpg** → City miles per gallon
- **highway-mpg** → Highway miles per gallon
- **price** → The price of the car (the label variable)

Exercise instructions:

Use the **CarPricePrediction.csv** file for this exercise.

Data preparation:

1. Check for missing values in the dataset.
2. Check for duplicate rows in the dataset.
3. In case you found any of those, remove them from the df.
4. Convert categorical features to numerical form using the get_dummies() method.

Simple Decision Tree Machine Learning:

1. Use your preprocessing dataset.
2. Apply train / test split with a test size of 30% and a random seed of 42.
3. Apply a Decision Tree Regressor to predict the `price`.
4. Plot the decision tree your model created, make sure every leaf in the tree has gini impurity of 0.
5. Determine according to the tree you plot if there is a risk for overfitting.
6. Print the features level of importance according to the model decision.
7. Predict the car prices of the test set.
8. Evaluate your model using MAE, MSE, and RMSE.
9. Plot the true vs. predicted prices.

Optimal Decision Tree Machine Learning:

1. Use your preprocessing dataset.
2. Apply the train/test split with a test size of 30% and a random seed of 42.
3. Use Grid Search to find the optimal `max_depth`, `min_samples_split` and ‘max_leaf_nodes’ from specified ranges at your choice.
   Choose ranges that make sense according to the number of features of the data set and according to the size of the training dataset.
4. Use MSE as the score metric to evaluate the optimal parameters.
5. Print the optimal parameters your Grid Search provided.
6. Train your Decision Tree Regressor with the optimal parameters found.
7. Evaluate your model using MAE, MSE, and RMSE.
8. Check for improvement from the simple Decision Tree model.
9. Plot the true vs. predicted prices.

Optimal Random Forest Machine Learning:

1. Use your preprocessing dataset.
2. Ensure categorical features are converted to numerical.
3. Apply the train / test split with a test size of 30% and a random seed of 42.
4. Use Grid Search to find the optimal `n_estimators`, `max_features`, `bootstrapping` and `min_samples_split` from specified ranges.
5. Choose reasonable ranges for these hyperparameters based on the dataset's characteristics.
6. Provide the Grid Search a Random Forest model.
7. Train your Decision Tree Regressor with the optimal parameters found.
8. Use MSE as the score metric to evaluate the optimal parameters.
9. Print the optimal parameters your Grid Search provided.
10. Train your Random Forest Regressor with the optimal parameters found.
11. In case your Grid Search chose to use bootstrapping add also OOB scoring to your Random Forest model.
12. Evaluate your model using MAE, MSE, and RMSE.
13. Check for improvement from the Simple Decision Tree model your trained previously
14. Plot the true vs. predicted prices.

Model Deployment:

1. Train your final Random Forest model on the entire dataset using the optimal parameters.
2. Export your final model into a joblib file.
3. Ensure that you also export other relevant preprocessing instances such as the standard scaler and one-hot encoder.
4. Import and Load → Import your final model and the preprocessing instances from the joblib files back to your working area when needed for deployment.

Model New Predictions:

Use the following code to get 2 new data points and predict their car price according to your optimal Decision Tree model and optimal Random Forest model.

```
new_data = pd.DataFrame({
'symboling': [1, 2],
'normalized-losses': [95, 84],
'make': ['toyota', 'honda'],
'fuel-type': ['gas', 'gas'],
'aspiration': ['std', 'std'],
'num-of-doors': ['four', 'four'],
'body-style': ['sedan', 'hatchback'],
'drive-wheels': ['fwd', 'fwd'],
'engine-location': ['front', 'front'],
'wheel-base': [98.0, 96.5],
'length': [176.2, 167.0],
'width': [66.5, 65.4],
'height': [54.3, 52.6],
'curb-weight': [2579, 2204],
'engine-type': ['ohc', 'ohc'],
'num-of-cylinders': ['four', 'four'],
'engine-size': [108, 97],
'fuel-system': ['mpfi', '2bbl'],
'bore': [3.50, 3.19],
'stroke': [2.80, 3.03],
'compression-ratio': [8.8, 9.6],
'horsepower': [75, 76],
'peak-rpm': [5000, 6000],
'city-mpg': [30, 30],
'highway-mpg': [38, 34]
})
```