14 Random Forest Models in Supervised Learning

Nandi Christmas

In [3]:
```
import pandas as pd
import numpy as np
df = pd.read_csv(
    filepath_or_buffer = "C:\output\cars_prepared.csv",
    engine = 'pyarrow',
)
list_reduced = [
    'symboling',
    'doornumber',
    'wheelbase',
    'carlength',
    'carwidth',
    'carheight',
    'curbweight',
    'cylindernumber',
    'enginesize',
    'boreratio',
    'horsepower',
    'carlength_n_carwidth',
    'horsepower',
    'peakrpm',
    'citympg',
    'highwaympg',
    'CarName_1_low',
    'drivewheel_1_low',
    'enginelocation_3_high',
    'aspiration_3_high',
    'fuelsystem_3_high',
    'enginetype_2_moderate',
    'fueltype_2_moderate',
    'carbody_1_low'
    ]
X = df[list_reduced]
y = df['price']
X_Train = df[list_reduced].loc[df['TrainTest'] == 'Train']
y_Train = df['price'].loc[df['TrainTest'] == 'Train']
df['price'].hist()
```

```
<>:4: SyntaxWarning: invalid escape sequence '\n'
<>:4: SyntaxWarning: invalid escape sequence '\n'
C:\Users\nandi\AppData\Local\Temp\ipykernel_9224128680756 2.py:4: SyntaxWarning: invalid escape sequence '\n'
  filepath_or_buffer = "C:\output\cars_prepared.csv",
```

Out[3]:
```
<Axes: >
```

In [4]:
```
list_max_depth = [9,17,35,None]
list_min_samples_leaf = [1,4,20]
min_samples_split = [2,8,60]

from sklearn.ensemble import RandomForestRegressor
list_RandomForestRegressor = [
    RandomForestRegressor(
        max_depth = i,
        min_samples_leaf = k,
        min_samples_split = l,
        oob_score = True,
        random_state = 823
    )
    for i in list_max_depth for k in list_min_samples_leaf for l in min_samples_split
]
```

In [5]:
```
list_fit = [model.fit(
    X = X_Train,
    y = y_Train
) for model in list_RandomForestRegressor]
```

In [6]:
```
list_predict = [fit.predict(X = X) for fit in list_fit]
df_predict = pd.DataFrame(list_predict).T
df_predict.head()
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	...	26	27	28	29	30	31	32	33	34	35
0	9.456835	9.442816	9.366929	9.428075	9.428075	9.366929	9.379228	9.379228	9.379228	9.462568	...	9.379228	9.462568	9.442816	9.366929	9.428075	9.428075	9.366929	9.379228	9.379228	9.379228
1	9.456835	9.442816	9.366929	9.428075	9.428075	9.366929	9.379228	9.379228	9.379228	9.462568	...	9.379228	9.462568	9.442816	9.366929	9.428075	9.428075	9.366929	9.379228	9.379228	9.379228
2	9.769909	9.785934	9.382721	9.776884	9.776884	9.382721	9.379228	9.379228	9.379228	9.767959	...	9.379228	9.767959	9.785934	9.382721	9.776884	9.776884	9.382721	9.379228	9.379228	9.379228
3	9.209384	9.199297	9.359504	9.193575	9.193575	9.359504	9.357774	9.357774	9.357774	9.200069	...	9.357774	9.200069	9.199297	9.359504	9.193575	9.193575	9.359504	9.357774	9.357774	9.357774
4	9.717993	9.742607	9.382721	9.729553	9.729553	9.382721	9.379228	9.379228	9.379228	9.707275	...	9.379228	9.707275	9.742607	9.382721	9.729553	9.729553	9.382721	9.379228	9.379228	9.379228

5 rows × 36 columns

In [7]:
```
from sklearn.metrics import mean_squared_error
list_mean_squared_error = [[i,k,mean_squared_error(
    y_true = df['price'].loc[df['TrainTest'] == k],
    y_pred = df_predict.loc[df['TrainTest'] == k,i]
)] for i in range(df_predict.shape[1]) for k in ['Train','Validation','Test']]
df_mean_squared_error = pd.DataFrame(list_mean_squared_error)
df_mean_squared_error.columns = ['model','TrainTest','mean_squared_error']
df_mean_squared_error = df_mean_squared_error.pivot(
    index='model',
    columns='TrainTest',
    values='mean_squared_error'
)
df_mean_squared_error['RandomForestRegressor'] = list_RandomForestRegressor
df_mean_squared_error['oob_score_'] = [fit.oob_score_ for fit in list_fit]
df_mean_squared_error = df_mean_squared_error.sort_values(['Validation','Test','Train'])[['RandomForestRegressor','Test','Validation','Train','oob_score_']]
```

In [8]:
```
df_mean_squared_error
```

Out[8]:

TrainTest	RandomForestRegressor	Test	Validation	Train	oob_score_
model					
0	(DecisionTreeRegressor(max_depth=8, max_featur...	0.038136	0.020986	0.002514	0.918616
9	(DecisionTreeRegressor(max_depth=17, max_featu...	0.038753	0.021074	0.002546	0.916745
18	(DecisionTreeRegressor(max_depth=35, max_featu...	0.038753	0.021074	0.002546	0.916745
27	(DecisionTreeRegressor(max_features=1.0, rando...	0.038753	0.021074	0.002546	0.916745
10	(DecisionTreeRegressor(max_depth=17, max_featu...	0.039618	0.021209	0.005778	0.919697
19	(DecisionTreeRegressor(max_depth=35, max_featu...	0.039618	0.021209	0.005778	0.919697
28	(DecisionTreeRegressor(max_features=1.0, min_s...	0.039618	0.021209	0.005778	0.919697
1	(DecisionTreeRegressor(max_depth=8, max_featur...	0.039647	0.021217	0.005795	0.919193
3	(DecisionTreeRegressor(max_depth=8, max_featur...	0.041534	0.022548	0.007775	0.912795
4	(DecisionTreeRegressor(max_depth=8, max_featur...	0.041534	0.022548	0.007775	0.912795
12	(DecisionTreeRegressor(max_depth=17, max_featu...	0.041534	0.022548	0.007775	0.912795
13	(DecisionTreeRegressor(max_depth=17, max_featu...	0.041534	0.022548	0.007775	0.912795
21	(DecisionTreeRegressor(max_depth=35, max_featu...	0.041534	0.022548	0.007775	0.912795
22	(DecisionTreeRegressor(max_depth=35, max_featu...	0.041534	0.022548	0.007775	0.912795
30	(DecisionTreeRegressor(max_features=1.0, min_s...	0.041534	0.022548	0.007775	0.912795
31	(DecisionTreeRegressor(max_features=1.0, min_s...	0.041534	0.022548	0.007775	0.912795
2	(DecisionTreeRegressor(max_depth=8, max_featur...	0.308395	0.267519	0.227239	0.001252
5	(DecisionTreeRegressor(max_depth=8, max_featur...	0.308395	0.267519	0.227239	0.001252
11	(DecisionTreeRegressor(max_depth=17, max_featu...	0.308395	0.267519	0.227239	0.001252
14	(DecisionTreeRegressor(max_depth=17, max_featu...	0.308395	0.267519	0.227239	0.001252
20	(DecisionTreeRegressor(max_depth=35, max_featu...	0.308395	0.267519	0.227239	0.001252
23	(DecisionTreeRegressor(max_depth=35, max_featu...	0.308395	0.267519	0.227239	0.001252
29	(DecisionTreeRegressor(max_features=1.0, min_s...	0.308395	0.267519	0.227239	0.001252
32	(DecisionTreeRegressor(max_features=1.0, min_s...	0.308395	0.267519	0.227239	0.001252
6	(DecisionTreeRegressor(max_depth=8, max_featur...	0.309546	0.267601	0.227755	0.001411
7	(DecisionTreeRegressor(max_depth=8, max_featur...	0.309546	0.267601	0.227755	0.001411
8	(DecisionTreeRegressor(max_depth=8, max_featur...	0.309546	0.267601	0.227755	0.001411
15	(DecisionTreeRegressor(max_depth=17, max_featu...	0.309546	0.267601	0.227755	0.001411
16	(DecisionTreeRegressor(max_depth=17, max_featu...	0.309546	0.267601	0.227755	0.001411
17	(DecisionTreeRegressor(max_depth=17, max_featu...	0.309546	0.267601	0.227755	0.001411
24	(DecisionTreeRegressor(max_depth=35, max_featu...	0.309546	0.267601	0.227755	0.001411
25	(DecisionTreeRegressor(max_depth=35, max_featu...	0.309546	0.267601	0.227755	0.001411
26	(DecisionTreeRegressor(max_depth=35, max_featu...	0.309546	0.267601	0.227755	0.001411
33	(DecisionTreeRegressor(max_features=1.0, min_s...	0.309546	0.267601	0.227755	0.001411
34	(DecisionTreeRegressor(max_features=1.0, min_s...	0.309546	0.267601	0.227755	0.001411
35	(DecisionTreeRegressor(max_features=1.0, min_s...	0.309546	0.267601	0.227755	0.001411

In [9]:
```
pd.DataFrame({
    'feature_names_in_' : list_RandomForestRegressor[9].feature_names_in_,
    'feature_importances_' : list_RandomForestRegressor[9].feature_importances_
}).sort_values('feature_importances_')
```

Out[9]:

	feature_names_in_	feature_importances_
18	enginelocation_3_high	0.000004
1	doornumber	0.000235
17	drivewheel_1_low	0.000303
19	aspiration_3_high	0.000578
21	enginetype_2_moderate	0.000675
23	carbody_1_low	0.000721
0	symboling	0.000940
22	fueltype_2_moderate	0.001274
13	peakrpm	0.002083
20	fuelsystem_3_high	0.002487
5	carheight	0.003156
11	compressionratio	0.003407
10	stroke	0.004153
2	wheelbase	0.005269
9	boreratio	0.005309
12	horsepower	0.005566
3	carlength	0.013245
16	CarName_1_low	0.015806
4	carwidth	0.018519
7	cylindernumber	0.025111
14	citympg	0.064549
8	enginesize	0.189791
15	highwaympg	0.193264
6	curbweight	0.442558

In [10]:
```
dir(list_RandomForestRegressor[9])
```

Out[10]:
```
['__abstractmethods__',
 '__annotations__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__firstlineno__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setstate__',
 '__sizeof__',
 '__slots__',
 '__str__',
 '__static_attributes__',
 '__subclasshook__',
 '__weakref__',
 '_abc_impl',
 '_build_request_for_signature',
 '_check_feature_names',
 '_check_n_features',
 '_compute_oob_predictions',
 '_compute_partial_dependence_recursion',
 '_doc_link_module',
 '_doc_link_template',
 '_doc_link_url_param_generator',
 '_estimator_type',
 '_get_default_requests',
 '_get_doc_link',
 '_get_estimators_indices',
 '_get_metadata_request',
 '_get_oob_predictions',
 '_get_params_names',
 '_get_tags',
 '_make_estimator',
 '_more_tags',
 '_n_samples',
 '_n_samples_bootstrap',
 '_parameter_constraints',
 '_repr_html_',
 '_repr_html_inner',
 '_repr_mimebundle_',
 '_required_parameters',
 '_set_oob_score_and_attributes',
 '_validate_X_predict',
 '_validate_data',
 '_validate_estimator',
 '_validate_params',
 '_validate_y_class_weight',
 'apply',
 'base_',
 'ccp_alpha',
 'class_weight',
 'criterion',
 'decision_path',
 'estimator',
 'estimator_',
 'estimator_params',
 'estimators',
 'estimators_samples_',
 'feature_importances_',
 'feature_names_in_',
 'fit',
 'get_metadata_routing',
 'get_params',
 'max_depth',
 'max_features',
 'max_leaf_nodes',
 'max_samples',
 'min_impurity_decrease',
 'min_samples_leaf',
 'min_samples_split',
 'min_weight_fraction_leaf',
 'monotonic_cst',
 'n_estimators',
 'n_features_in_',
 'n_jobs',
 'n_outputs_',
 'oob_prediction_',
 'oob_score',
 'oob_score_',
 'predict',
 'random_state',
 'score',
 'set_fit_request',
 'set_params',
 'set_score_request',
 'verbose',
 'warm_start']
```

In [111]:
```
list_RandomForestRegressor[9].oob_score_
```

Out[111]:
```
0.9167448936748511
```