Assignment 2

assignment2.csv is the data for you to do analysis on. It is the data to predict cars' prices.

1. There are three numerical features: [length, width, height], two categorical features: [make, drive-wheels]
2. What you need to do is using polynomial and categorical regression method as we showed in tutorial to build some models
3. Specifically, build a model with the following features:
   - `width`
   - degree of 2 polynomial of `length` *
   - degree of 3 polynomial of `height` *
   - two categorial features `make` and `drive-wheels`
4. consider the whole data set as training set, fit the model.
5. compute the RMSE and R^2 of this model.

*You may use the function `create_poly_feature()` defined as follows to generate the polynomials.

Below is the guideline:

1. Read the data, call it data_original
2. Target = 'price'
3. Fix the features, e.g., features_numerical = ['A', 'B', 'C', 'D'], features_category=['E','F'].
4. Preprocess the data
   - add one-hot encoding for categorical to `data`
   - add polynomial to `data`
5. Fit to model, `model.fit(data.drop[target], data[target])`
6. Predict the model by `data`

1. Import a few tools we need

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model, metrics, model_selection
```

2. Play with the data

2.1 read the data by pandas

In [2]:
```
assignment_origin = pd.read_csv('assignment2.csv')
```

In [3]:
```
assignment_origin
```

Out[3]:

	make	drive-wheels	length	width	height	price
0	alfa-romero	rwd	168.8	64.1	48.8	16500
1	alfa-romero	rwd	171.2	65.5	52.4	16500
2	audi	fwd	176.6	66.2	54.3	13950
3	audi	4wd	176.6	66.4	54.3	17450
4	audi	fwd	177.3	66.3	53.1	15250
...	...	...	...	...	...	...
175	volvo	rwd	188.8	67.2	57.5	18950
176	volvo	rwd	188.8	68.9	55.5	16845
177	volvo	rwd	188.8	68.8	55.5	19045
178	volvo	rwd	188.8	68.9	55.5	21485
179	volvo	rwd	188.8	68.9	55.5	22625

180 rows × 6 columns

In [5]:
```
# split features into groups
target = ['price']
features_numerical = ['length','width','height']
features_category = ['make','drive-wheels']

assignment = assignment_origin[ target + features_numerical + features_category]
```

2.2 create additional features for Polynomial part

In [6]:
```
# This function returns a new dataframe,
# which contains all powers of features from 2 to what you like.

def create_poly_feature(df, feature, degree):
    result = pd.DataFrame()
    if feature in df.columns:
        # loop over the degrees:
        for power in range(2,  degree+1):
            # first we'll give the column a name:
            name = feature + '_power_' + str(power)

            result[name] = df[feature].astype(float) ** power
        return result
    else:
        return print("Please select a feature in this df!")
```

In [7]:
```
# create new features
poly_feature_length = create_poly_feature(assignment, 'length' , 2)
poly_feature_height = create_poly_feature(assignment, 'height', 3)
```

2.3 create additional features for Category part

In [8]:
```
# This function returns a new dataframe,
# the categorical feature will be replaced by its onehot transformation

def onehot_encoder(df, feature):
    result = pd.DataFrame()
    if feature in df.columns:
        result = pd.get_dummies(df, columns=[feature])
        return result
    else:
        return print("Please select a feature in this df!")
```

In [9]:
```
# create new features
category_feature_make = onehot_encoder(assignment, 'make')
category_feature_makeANDdrive = onehot_encoder(category_feature_make, 'drive-wheels')
```

Try to figure out: what is the difference between the two functions `onehot_encoder()` and `create_poly_feature()`

Answer: `onehot_encoder()` will return a dataframe with all other features, except a categorical one we manipulate. And this categorical feature will be replaced by its onehot transformation.

2.4 concatenate all dataframes together

In [10]:
```
assignment_processed = pd.concat([category_feature_makeANDdrive,
                                  poly_feature_height,
                                  poly_feature_length],
                                  axis = 1)
```

2.5 split target and features

In [11]:
```
X = assignment_processed.drop('price',axis=1)
Y = assignment_processed['price']
```

3. create the linear model

In [12]:
```
model = linear_model.LinearRegression()

model.fit(X,Y)
```

Out[12]:
```
LinearRegression()
```

4. analysis

In [13]:
```
# prediction
Y_fit = model.predict(X)

print("square root of mean squared error: $%.2f"
      % np.sqrt(metrics.mean_squared_error(Y, Y_fit)))

# Explained R2 score: 1 is perfect prediction
print('R2 score: %.2f' % metrics.r2_score(Y, Y_fit))
```

```
square root of mean squared error: $2392.45
R2 score: 0.91
```