1 Executive Summary

When we have continuous-by-continuous interactions in linear regression, it is impossible to directly interpret the coefficients on the interactions. Generalized Additive Models(GAMs) extend generalized linear models by allowing non-linear functions of features while maintaining additivity. Since the model is additive, it is easy to examine the effect of each X_i on Y individually while holding all other predictors constant. The result is a very flexible model, where it is easy to incorporate prior knowledge and control overfitting. In this notebook, I will use **pyGAM** pachage in python which allow me to fit my linear model in the presence of interaction terms.

2 Introduction

I plan to build a linear regression model in python for the Boston house-price data which is taken from the StatLib library which is maintained at Carnegie Mellon University. Additionally, I am interested to capture the interacton terms for this model.

In [26]:
```
# importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [47]:
```
from sklearn.datasets import load_boston
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
target_df = pd.Series(boston.target)
df.head()
```

Out[47]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [194]:
```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (https://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [48]:
```
df.shape
```

Out[48]: (506, 13)

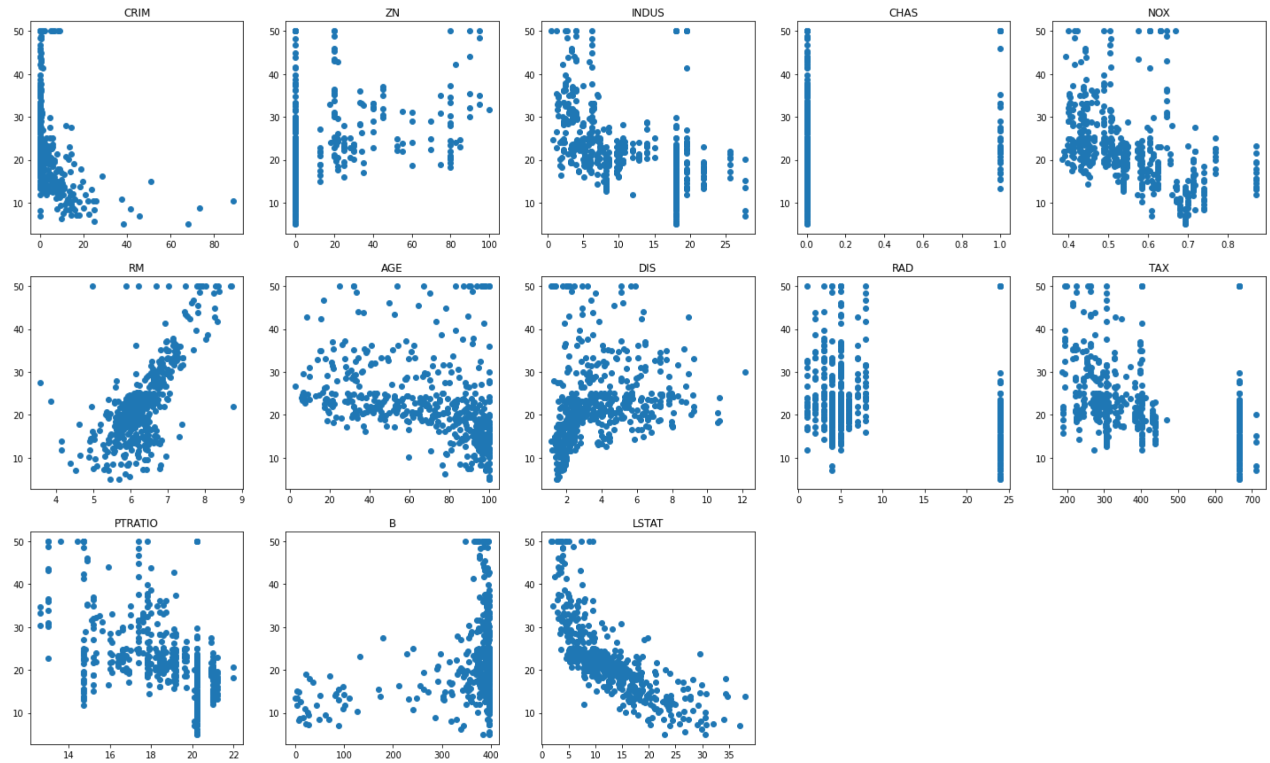
In [49]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

In [202]:

```
n_cols = 5
fig, axes = plt.subplots(nrows=3,
                         ncols=n_cols,
                         figsize=(25,15))
j=0
for _, ax in zip(letter, axes.flatten()):
    ax.scatter(boston.data[:, j], boston.target)
    ax.set_title(boston.feature_names[j])
    j=j+1
fig.delaxes(axes[2,3])
fig.delaxes(axes[2,4])
```



The response variable is "mpg", "cyl" and "hp" represents the number of Cylinder and Horsepower Of a car. Let us visualize the relationship between the Weight, horsepower and Miles per Gallon which is also conditioned on a third variable "cyl" and plot the levels of the Culinder in different colors.

3 Building the Regression Model

3.1 PyGAM

In [179]:

```
from pygam import LinearGAM, s, f, te, l
```

3.1.1 Direct Effect

In [173]:

```
X = df
y = target_df
gam = LinearGAM().fit(X, y)
```

When we fit a LinearGAM without without providing the features it automatically creates a spline with 20 functions for each predictor. They are represented as S(feature index) in the model summary. In pyGAM, we specify the functional form using terms:

- l() linear terms: for terms like Xi
- s() spline terms
- f() factor terms
- te() tensor products
- intercept.

Note: GAM(..., intercept=True) so models include an intercept by default.

We can specify the number of splines for each feature by ourselves. pyGAM can also fit interactions using tensor products via te().We can treat a predictor also as a factor fpr example if it is approximately continuous, but only takes on specific number of discrete values like 10 or 3.

In [63]:

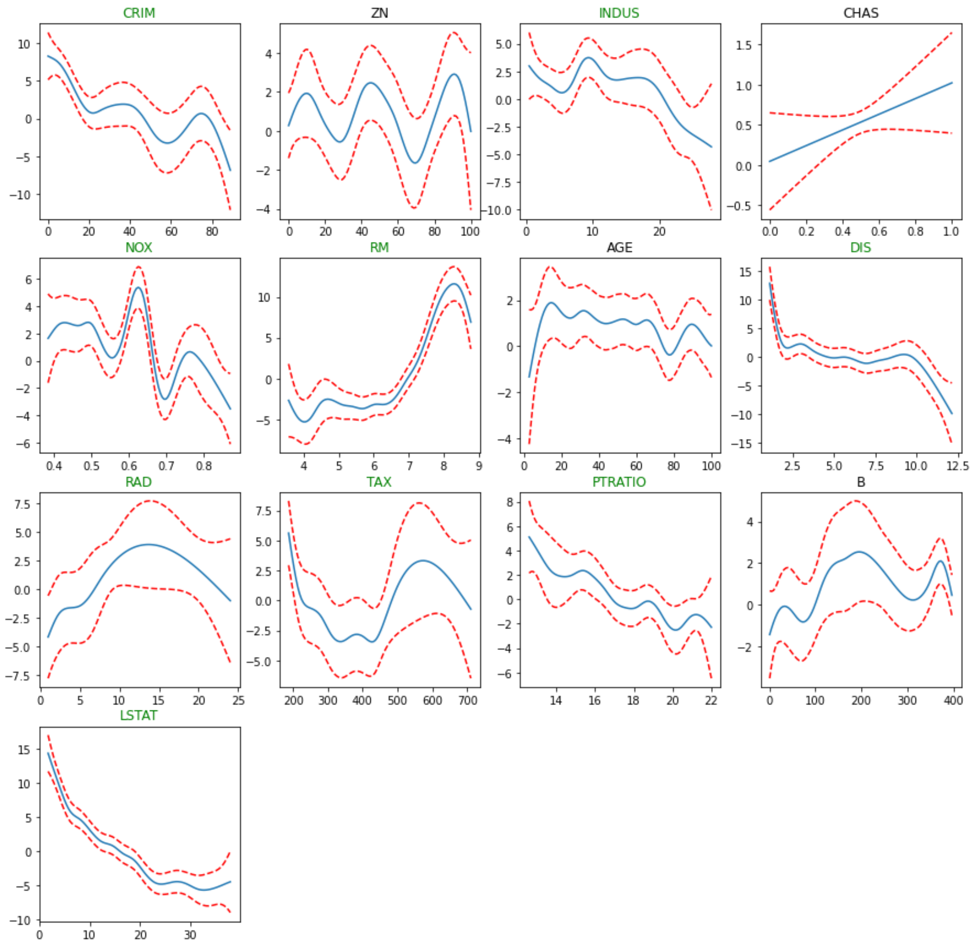
```
gam.summary()
```

```
LinearGAM
=============================================== ==========================================================
Distribution:                        NormalDist Effective DoF:                                    103.2423
Link Function:                     IdentityLink Log Likelihood:                                 -1589.7653
Number of Samples:                          506 AIC:                                             3388.0152
                                                AICc:                                            3442.7649
                                                GCV:                                               13.7683
                                                Scale:                                              8.8269
                                                Pseudo R-Squared:                                   0.9168
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           11.1         2.20e-11     ***
s(1)                              [0.6]                20           12.8         8.15e-02     .
s(2)                              [0.6]                20           13.5         2.59e-03     **
s(3)                              [0.6]                20           3.8          2.76e-01
s(4)                              [0.6]                20           11.4         1.11e-16     ***
s(5)                              [0.6]                20           10.1         1.11e-16     ***
s(6)                              [0.6]                20           10.4         8.22e-01
s(7)                              [0.6]                20           8.5          4.44e-16     ***
s(8)                              [0.6]                20           3.5          5.96e-03     **
s(9)                              [0.6]                20           3.4          1.33e-09     ***
s(10)                             [0.6]                20           1.8          3.26e-03     **
s(11)                             [0.6]                20           6.4          6.25e-02     .
s(12)                             [0.6]                20           6.5          1.11e-16     ***
intercept                                              1            0.0          2.23e-13     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

According to this model, the impact of 9 predictors are significant. We will consider only those predictors in the future version of the model.

In [165]:

```python
letter = "ABCDEFGHIJKLM"
subtitles = ['CRIM','INDUS','NOX', 'RM','DIS', 'RAD', 'TAX','PTRATIO','LSTAT']
titles =['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
        'PTRATIO', 'B', 'LSTAT']
n_cols = 4
fig, axes = plt.subplots(nrows=4,
                         ncols=n_cols,
                         figsize=(15,15))
j=0
for _, ax in zip(letter, axes.flatten()):
    XX = gam.generate_X_grid(term=j)
    ax.plot(XX[:, j], gam.partial_dependence(term=j, X=XX))
    ax.plot(XX[:, j], gam.partial_dependence(term=j, X=XX, width=.95)[1], c='r', ls='--')
    if titles[j] in subtitles:
        ax.set_title(titles[j], color="Green")
    else:
        ax.set_title(titles[j])
    j=j+1
fig.delaxes(axes[3,1])
fig.delaxes(axes[3,2])
fig.delaxes(axes[3,3])
```



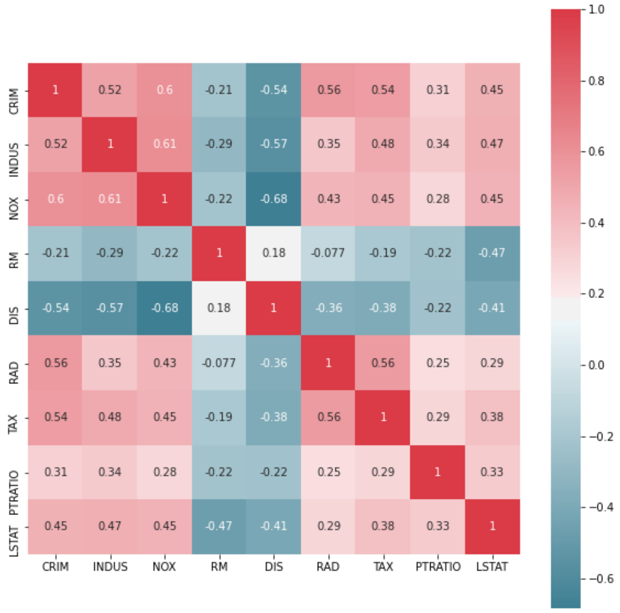
The plots shows that the relationships between our predictors and target variables obviously are not linear. Those which are important are highlighted with green titles.

3.1.2 Interaction Effects

3.1.2.1 Correlation Heatmaps

In [171]:
```
import seaborn as sns
f, ax = plt.subplots(figsize=(10, 10))
corr = X[['CRIM','INDUS','NOX', 'RM','DIS', 'RAD', 'TAX','PTRATIO','LSTAT']].corr(method ='kendall')
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, annot=True, ax=ax)
```

Out[171]: <AxesSubplot:>



I have decided to consider the following correlations as the interatcion terms in my model ('DIS','NOX') ('NOX', 'CRIM'),('NOX','INDUS'), ('TAX','CRIM'), ('TAX', 'RAD')

- CRIM shows per capita crime rate by town
- INDUS shows proportion of non-retail business acres per town
- NOX shows nitric oxides concentration (parts per 10 million)
- DIS shows weighted distances to five Boston employment centres
- RAD shows index of accessibility to radial highways
- TAX shows full-value property-tax rate per 10,000 US dollar

3.1.3 Main Effect

In [176]:
```
gam =LinearGAM().fit(X[['CRIM','INDUS','NOX', 'RM','DIS', 'RAD', 'TAX','PTRATIO','LSTAT']], y)
```

In [177]:
```
gam.summary()
```

```
LinearGAM
=============================================== ==========================================================
Distribution:                        NormalDist Effective DoF:                                     76.093
Link Function:                     IdentityLink Log Likelihood:                                -1610.7041
Number of Samples:                          506 AIC:                                            3375.5942
                                                AICc:                                           3403.7331
                                                GCV:                                              12.5268
                                                Scale:                                             9.1893
                                                Pseudo R-Squared:                                  0.9075
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           10.3         8.22e-14     ***
s(1)                              [0.6]                20           13.2         5.32e-03     **
s(2)                              [0.6]                20           11.5         1.11e-16     ***
s(3)                              [0.6]                20           10.0         1.11e-16     ***
s(4)                              [0.6]                20           10.3         1.11e-16     ***
s(5)                              [0.6]                20           4.4          9.05e-03     **
s(6)                              [0.6]                20           4.9          2.30e-10     ***
s(7)                              [0.6]                20           3.4          1.10e-07     ***
s(8)                              [0.6]                20           8.0          1.11e-16     ***
intercept                                              1            0.0          9.33e-15     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

3.1.4 Interaction Effect

In [184]:
```
gam =LinearGAM(s(0) + s(1) + s(2) + s(3) + s(4) + s(5) + s(6) + s(7) + s(8) + te(4,2) + te(2,0) + te(2,1)+ te(6,0) + te(6,5)).fit(X, y)
```

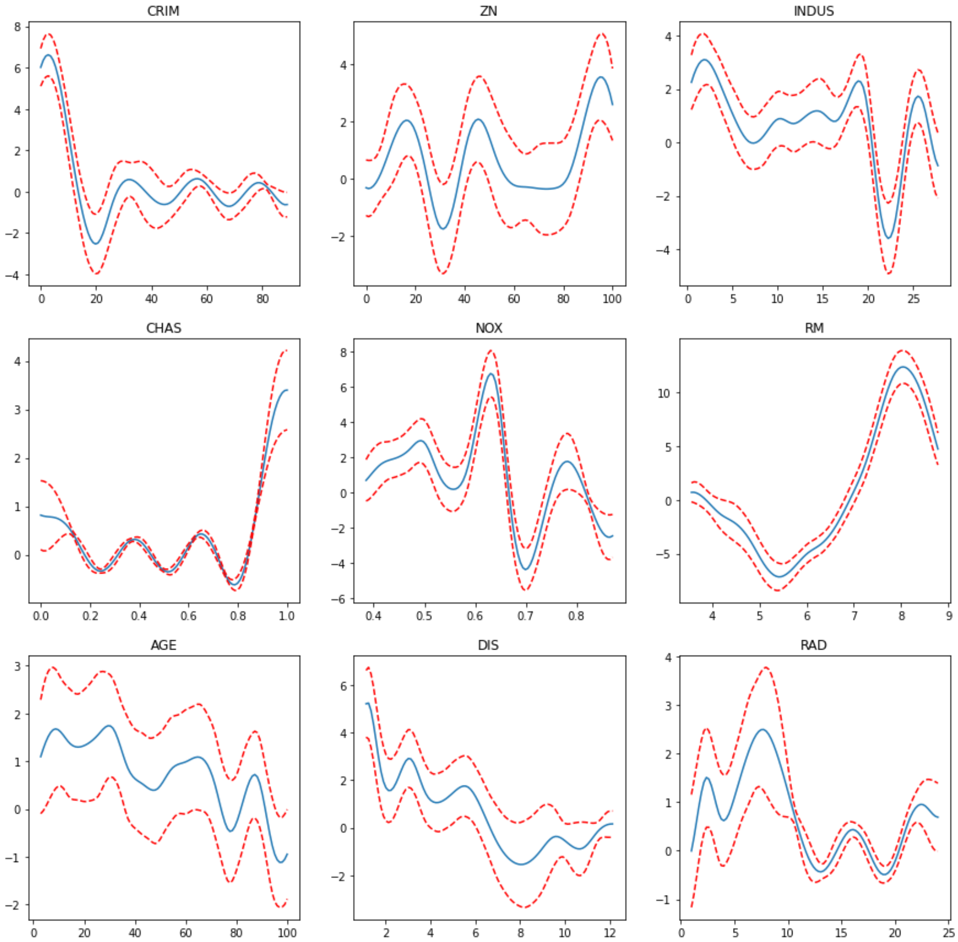
In [185]:
```
gam.summary()
```

```
LinearGAM
=============================================== ==========================================================
Distribution:                        NormalDist Effective DoF:                                    66.8278
Link Function:                     IdentityLink Log Likelihood:                                 -1877.023
Number of Samples:                          506 AIC:                                            3889.7017
                                                AICc:                                           3911.0591
                                                GCV:                                              20.7067
                                                Scale:                                            15.8507
                                                Pseudo R-Squared:                                   0.837
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20                        1.11e-16     ***
s(1)                              [0.6]                20                        1.11e-16     ***
s(2)                              [0.6]                20                        1.11e-16     ***
s(3)                              [0.6]                20                        1.11e-16     ***
s(4)                              [0.6]                20                        1.11e-16     ***
s(5)                              [0.6]                20                        1.11e-16     ***
s(6)                              [0.6]                20                        1.11e-16     ***
s(7)                              [0.6]                20                        1.11e-16     ***
s(8)                              [0.6]                20                        1.11e-16     ***
te(4, 2)                          [0.6 0.6]            100                       1.11e-16     ***
te(2, 0)                          [0.6 0.6]            100                       1.11e-16     ***
te(2, 1)                          [0.6 0.6]            100                       1.11e-16     ***
te(6, 0)                          [0.6 0.6]            100                       1.11e-16     ***
te(6, 5)                          [0.6 0.6]            100                       1.11e-16     ***
intercept                                              1                         1.11e-16     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

Obviously adding the interaction terms, made the model more complex. The model shows the significance of the predictors and their interactions.

In [191]:

```
n_cols = 3
fig, axes = plt.subplots(nrows=3,
                         ncols=n_cols,
                         figsize=(15,15))
j=0
for _, ax in zip(letter, axes.flatten()):
    XX = gam.generate_X_grid(term=j)
    ax.plot(XX[:, j], gam.partial_dependence(term=j, X=XX))
    ax.plot(XX[:, j], gam.partial_dependence(term=j, X=XX, width=.95)[1], c='r', ls='--')
    ax.set_title(titles[j])
    j=j+1
```



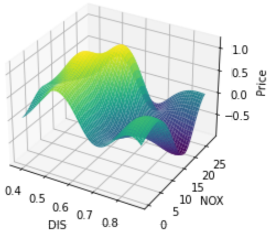
3.1.5 Visualizing the Interactions

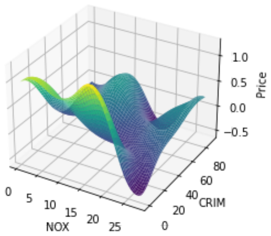
3.1.6 partial dependence plot

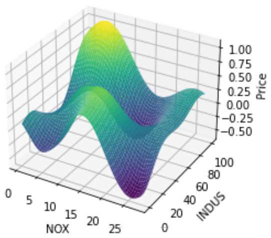
pyGAM supports partial dependence plot with matplotlib. The partial dependence for each term in a GAM can be visualized with a 95% confidence interval for the estimation function.

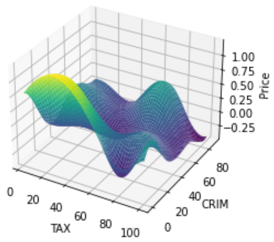
In [209]:

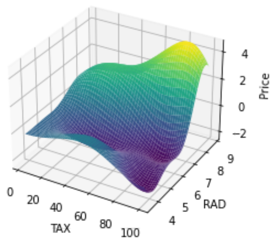
```python
import matplotlib.pyplot as plt
%matplotlib inline
plt.ion()
plt.rcParams['figure.figsize'] = (4, 4)
xlabelslist= ['DIS', 'NOX', 'NOX', 'TAX', 'TAX']
ylabelslist= ['NOX', 'CRIM', 'INDUS', 'CRIM', 'RAD']
j=0
for i in range(9, 14):
    fig = plt.figure()
    XX = gam.generate_X_grid(term=i, meshgrid=True)
    Z = gam.partial_dependence(term=i, X=XX, meshgrid=True)
    ax=plt.axes(projection='3d')
    xLabel = ax.set_xlabel(xlabelslist[j])
    yLabel = ax.set_ylabel(ylabelslist[j])
    zLabel = ax.set_zlabel('Price')
    ax.plot_surface(XX[0], XX[1], Z, cmap='viridis')
    j=j+1
```











The vertical height of the surface is the expected (predicted) value of y at each combination of predicted variables. The slope of the surface on each edge of the plot is a marginal effect. for example in the last plot, the slope on the lefthand face of the plot is the marginal effect of TAX when RAD==0 (or in its lowest value). Similarly, the slope on the righthand face of the plot is the marginal effect of RAD when TAX==1 (or in its largest value).