Saad Mufti

CS 539 - HW 2

Conceptual and Theoretical Questions

Q1

To minimize the error function, take the derivative with respect to $\mathbf{w}$ and set it to 0:

$$0 = -\frac{1}{2}\sum_{n=1}^{N} 2 \cdot r_n \left(t_n - w^T \cdot \phi(x_n)\right)\phi(x_n)^T$$

$$0 = -\sum_{n=1}^{N} r_n t_n \phi(x_n)^T - r_n w \phi(x_n)\phi(x_n)^T$$

$$0 = \sum_{n=1}^{N} r_n t_n \phi(x_n)^T - w\sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T$$

$$w\sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T = \sum_{n=1}^{N} r_n t_n \phi(x_n)^T$$

Take the inverse to isolate w

$$w = \left(\sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T\right)^{-1} \sum_{n=1}^{N} r_n t_n \phi(x_n)^T$$

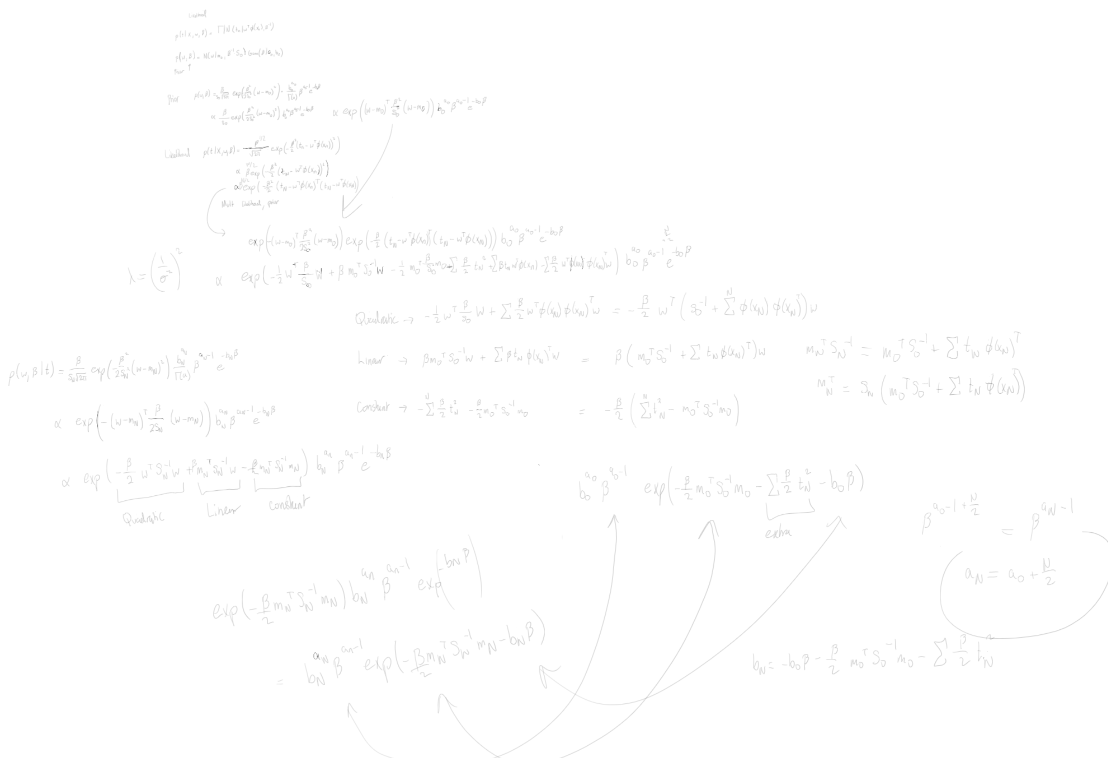
Because r_n is defined as the inverse of the variance of the nth data point, a large r_n indicates a small variance, indicating the high confidence in that data. A small weight would then of course indicate low certainty in that data with a high variance.

Alternatively, the weight r_n can simply be seen as how many times the nth data point has been observed, which is in line with the constraint $r_n > 0$, but is more fitting for integral values of r_n.

Q2

Using Bayes' rule, the relationship between the prior, likelihood, and posterior distributions is known to be:

$$p(\mathbf{w},\beta|\mathbf{t}) \propto p(\mathbf{w},\beta)\cdot p(\mathbf{t}|\mathbf{X},w,\beta)$$



Work is shown above

Q3

If a dataset is lineary separable, then its classes can be exactly distinguished with a linear decision boundary. The probability of a class C_1 in a dataset using logistic regression is

$$p(C_1|\phi) = \sigma(w^T\phi)$$

Because the dataset is linearly separable, data of one class x_a may fall one side of the decision boundary with $w^T\phi(x_a) > 0$, and another class denoted by x_b may be found on the other side $w^T\phi(x_b) < 0$.

If $|w| \to \infty$, then the sigmoid function of the posterior for class a $\sigma(w^T\phi) \to 1$, which thus maximizes the likelihood. This would also be true for finding class b since $\sigma(x) = 1 - \sigma(-x)$

Q4

We have

$$H = \nabla\nabla E(w) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T = \Phi^T R\Phi \qquad (4.97, \text{Bishop})$$

For a symmetric matrix A to be positive definite, then any real valued vector z can be multiplied by A to get all entries being positive:

$$z^T Az = A_{ij} > 0$$

Begin by multiplying the double gradient by z as above to prove it is positive definite:

$$z^T \sum_{n=1}^{N} (y_n(1-y_n)\phi_n\phi_n^T)z$$

$$\sum_{n=1}^{N} y_n(1-y_n)z^T\phi_n\phi_n^T z$$

Notice that $z^T\phi_n = (\phi_n^T z)^T$, which is also a dot product (scalar quantity), which allows us to square that term, making it always greater than or equal to zero:

$$\sum_{n=1}^{N} y_n(1-y_n)(\phi_n^T z)^2$$

Now, because $1 > y_n > 0$, since that is the range of the sigmoid function in logistic regression, then $y_n(1-y_n) > 0$ as well, meaning the terms are all positive.

As for proving that the function is concave, it has just been proven that the double gradient is always greater than 0, which ensures $\nabla\nabla E(w) > 0$, which means it is concave by definition.

Q5

Part a

The PMF of the exponential distribution family can be written as

$$p(x|\eta) = h(x)g(\eta)\exp\left\{\eta^T y(x)\right\} \qquad \text{(2.194, Bishop)}$$

The functions correspond to the following:

$$h(x) = x^{\alpha-1}$$
$$y(x) = x$$
$$\eta^T = -\beta$$
$$g(\eta) = g(\beta) = \beta^\alpha$$

Part b

In order to find the maximum likelihood, take the log for convenience:

$$\log(L) = \sum^N -\log(T(v)) + v\log(\frac{xt_i}{y_i}) - \log y_i - \frac{vt_i}{y_i}$$

Expanding and removing constant terms:

$$-(v+1)\sum^N \log y_i - v\sum^N \frac{t_i}{y_i}$$

To find the gradient ascent steps, take the gradient with respect to w_0 and w_1:

$$\frac{\partial \log(L)}{\partial w_0} = \sum^N -(v+1) + vt_i y_i$$
$$\frac{\partial \log(L)}{\partial w_1} = \sum^N x_i(-(v+1) + vt_i y_i)$$

This leads to the following update rules:

$$w_0^{(r+1)} \leftarrow w_0^r + \propto \nabla w_0^r$$
$$w_1^{(r+1)} \leftarrow w_1^r + \propto \nabla w_1^r$$

To determine if the likelihood is concave or not, take the gradient with respect to w_0 and w_1:

Q6

Part a

The likelihood function would be the probability of observing each t_n, given the parameters w and ϕ, over a normal distribution, so that:

$$L(t_1, t_2, \dots t_n | w, \phi) = \prod_{i=1}^{N} N(t_i | w^T \phi(x), \sigma\epsilon)$$

The Laplacian prior is defined as

$$P(w|b) = \frac{1}{2b} \exp\left(-\frac{|w|}{b}\right)$$

Multiply the prior and likelihood:

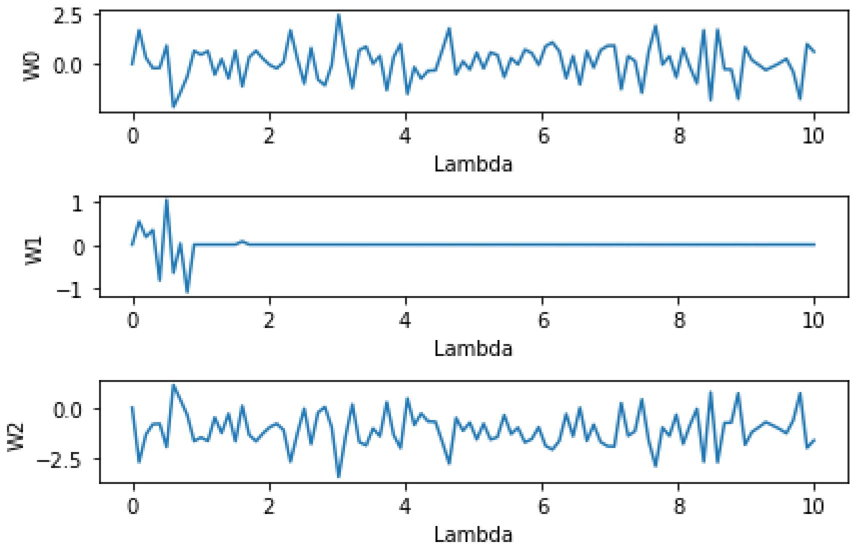
$$L(w|D, b) \propto L(t_1, t_2, \dots t_n | w, \phi) \cdot \frac{1}{2b} \exp\left(-\frac{|w|}{2b}\right)$$

Which follows the same general form as the lasso regularization formula:

$$-\frac{1}{2} \sum_i t_i - w^T \phi(x_i)^2 - \lambda \sum_j |w_i|$$

In [ ]:

```
# Part b
import numpy as np
import matplotlib.pyplot as plt

mu_ek = 0
sigma_ek = 0.1
ek = np.random.normal(loc=mu_ek, scale=np.sqrt(sigma_ek), size=(100,1))

mu_x = 0
sigma_x = 1
x_i = np.random.normal(loc=mu_x, scale=np.sqrt(sigma_x), size=(100,1))
t_i = 1 + (0.01 * x_i) - 2 *  np.multiply(x_i, x_i) + ek

lambda_1 = np.linspace(0, 10, num=100)
eps = 0.0001
phi_x = np.concatenate([np.ones((100, 1)), x_i, np.multiply(x_i,  x_i)], axis=
1)
alpha = 1e-4
ws = np.zeros((100, 3))
for l in np.arange(1, 100, step=1):
    t_lambda = lambda_1[l - 1]
    w_r = np.random.randn(3, 1)
    for iter in np.arange(1, 10000, step=1):
        err = t_i - np.matmul(phi_x, w_r) # 100, 1
        dw_0 = sum(err)

        d1 = 1 / max(eps, abs(w_r[1]))
        dw_1 = np.sum(phi_x[:, 1].T *err) - t_lambda * w_r[1] * d1

        d2 = 1 / max(eps, abs(w_r[2]))
        dw_2 = np.sum(phi_x[:, 2].T * err) - t_lambda * w_r[2] * d2

        w_r[0] = w_r[0] + alpha * dw_0
        w_r[1] = w_r[1] + alpha * dw_1
        w_r[2] = w_r[2] + alpha * dw_2
    ws[l,:] = w_r.reshape((3,))

fig_6, ax_6 = plt.subplots(3, 1)
ax_6[0].plot(lambda_1, ws[:, 0])
ax_6[0].set_ylabel("W0")
ax_6[0].set_xlabel("Lambda")

ax_6[1].plot(lambda_1, ws[:, 1])
ax_6[1].set_ylabel("W1")
ax_6[1].set_xlabel("Lambda")

ax_6[2].plot(lambda_1, ws[:, 2])
ax_6[2].set_ylabel("W2")
ax_6[2].set_xlabel("Lambda")
fig_6.tight_layout()
```



Application Questions

In [ ]:
```
# Imports

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from itertools import combinations
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression, LinearRegression, Bayesia
nRidge
from sklearn.metrics import mean_squared_error, RocCurveDisplay
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import cross_val_score
from scipy import stats
import pymc3 as pm
```

Linear Regression Problem (25 pts)

In [ ]:

```python
# Dataset init

dfr = pd.read_excel("Real estate valuation data set.xlsx")
dfr.rename(columns={
    "X1 transaction date": "X1",
    "X2 house age": "X2",
    "X3 distance to the nearest MRT station": "X3",
    "X4 number of convenience stores": "X4",
    "X5 latitude": "X5",
    "X6 longitude": "X6",
    "Y house price of unit area": "Y"
}, inplace=True)
print(f"Zeroing X1, transaction date, at {min(dfr['X1'])}")
dfr["X1"] = dfr["X1"] - min(dfr["X1"])
dfr.head()
```

```
Zeroing X1, transaction date, at 2012.6666667
```

Out[ ]:

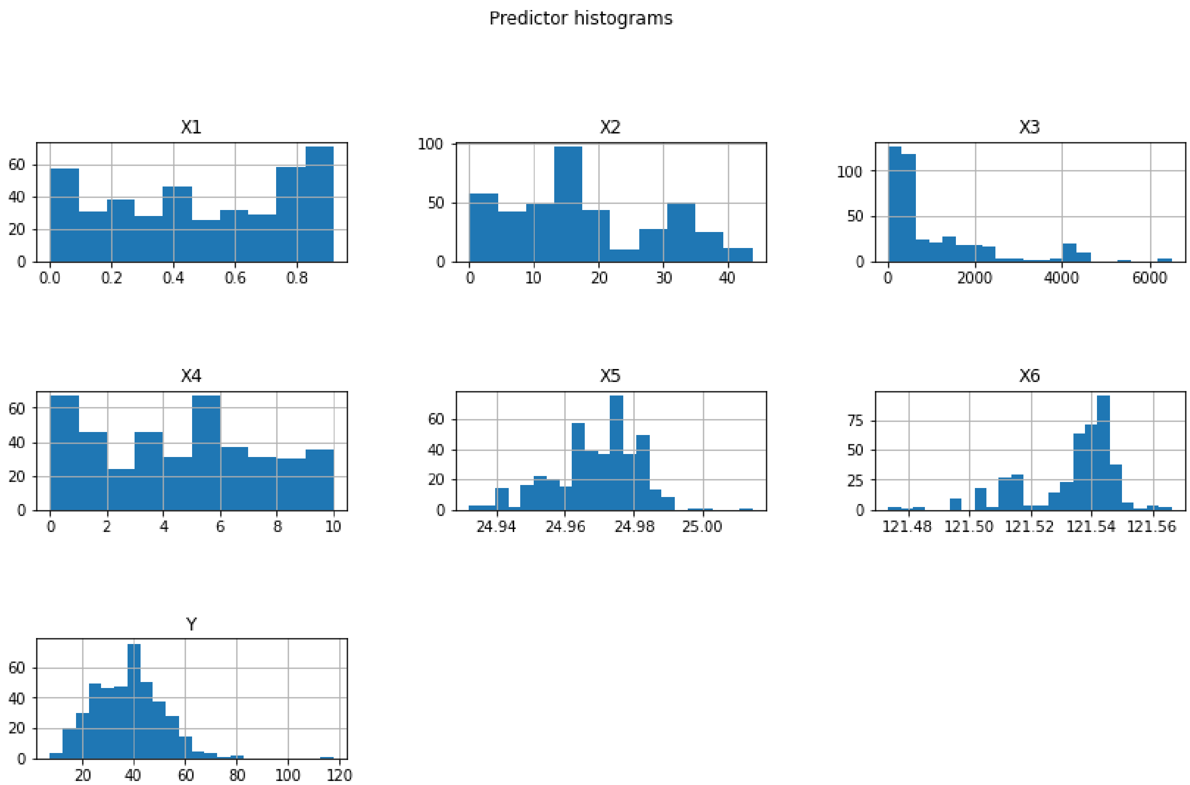
	No	X1	X2	X3	X4	X5	X6	Y
0	1	0.250000	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	0.250000	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	0.916667	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	0.833333	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	0.166667	5.0	390.56840	5	24.97937	121.54245	43.1

In [ ]:

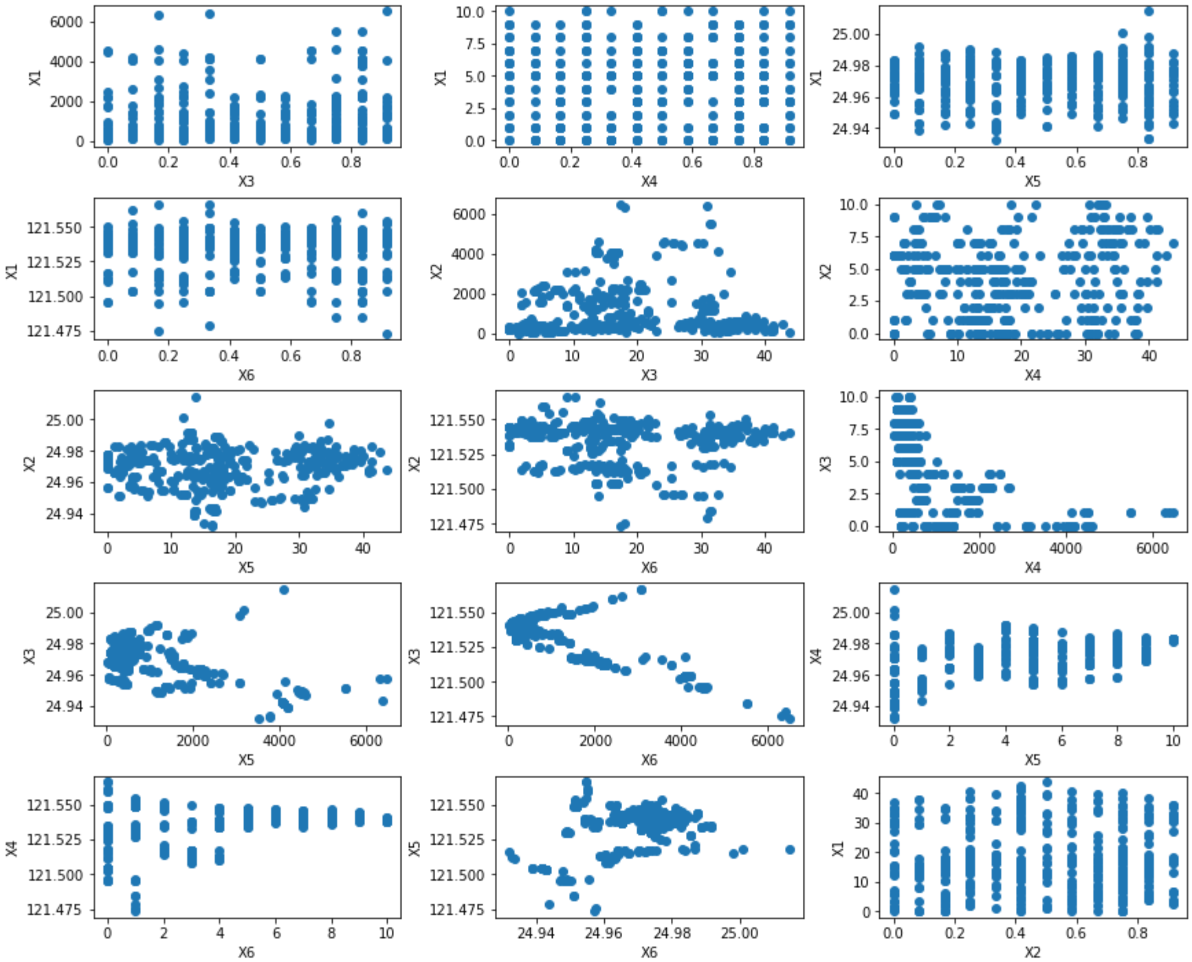
```python
# 1. Visualization

figure_hist, ax_hist = plt.subplots()
dfr.drop("No", axis=1).hist(ax=ax_hist, bins="auto")
figure_hist.suptitle("Predictor histograms")
figure_hist.tight_layout()
figure_hist.set_size_inches(12, 8)

fig_sc_preds, ax_preds = plt.subplots(nrows=5, ncols=3, figsize=(12, 10), cons
trained_layout=True)
fig_sc_resp, ax_resp = plt.subplots(nrows=2, ncols=3, figsize=(12, 10), constr
ained_layout=True)
combs = list(combinations(range(1,7), 2))

def lin_to_mat(cols, x):
    return (x // cols, x % cols)

for i in range(1, 7):
    ax_resp[lin_to_mat(3, i - 1)].scatter(dfr[f"X{i}"], dfr["Y"])
    ax_resp[lin_to_mat(3, i - 1)].set_xlabel(f"X{i}")
    ax_resp[lin_to_mat(3, i - 1)].set_ylabel("Y")
fig_sc_resp.suptitle("Scatter plots between predictors and response")

for i in range(len(combs)):
    ax_preds[lin_to_mat(3, i - 1)].scatter(dfr[f"X{combs[i][0]}"], dfr[f"X{com
bs[i][1]}"])
    ax_preds[lin_to_mat(3, i - 1)].set_ylabel(f"X{combs[i][0]}")
    ax_preds[lin_to_mat(3, i - 1)].set_xlabel(f"X{combs[i][1]}")
fig_sc_preds.suptitle("Scatter plots for predictors")

_="""
There appear to be some data points that differ from the rest of the predictor
data, such as in X3 and Y. X3 and X5 appear to have the strongest correlation
 with Y.
"""
```

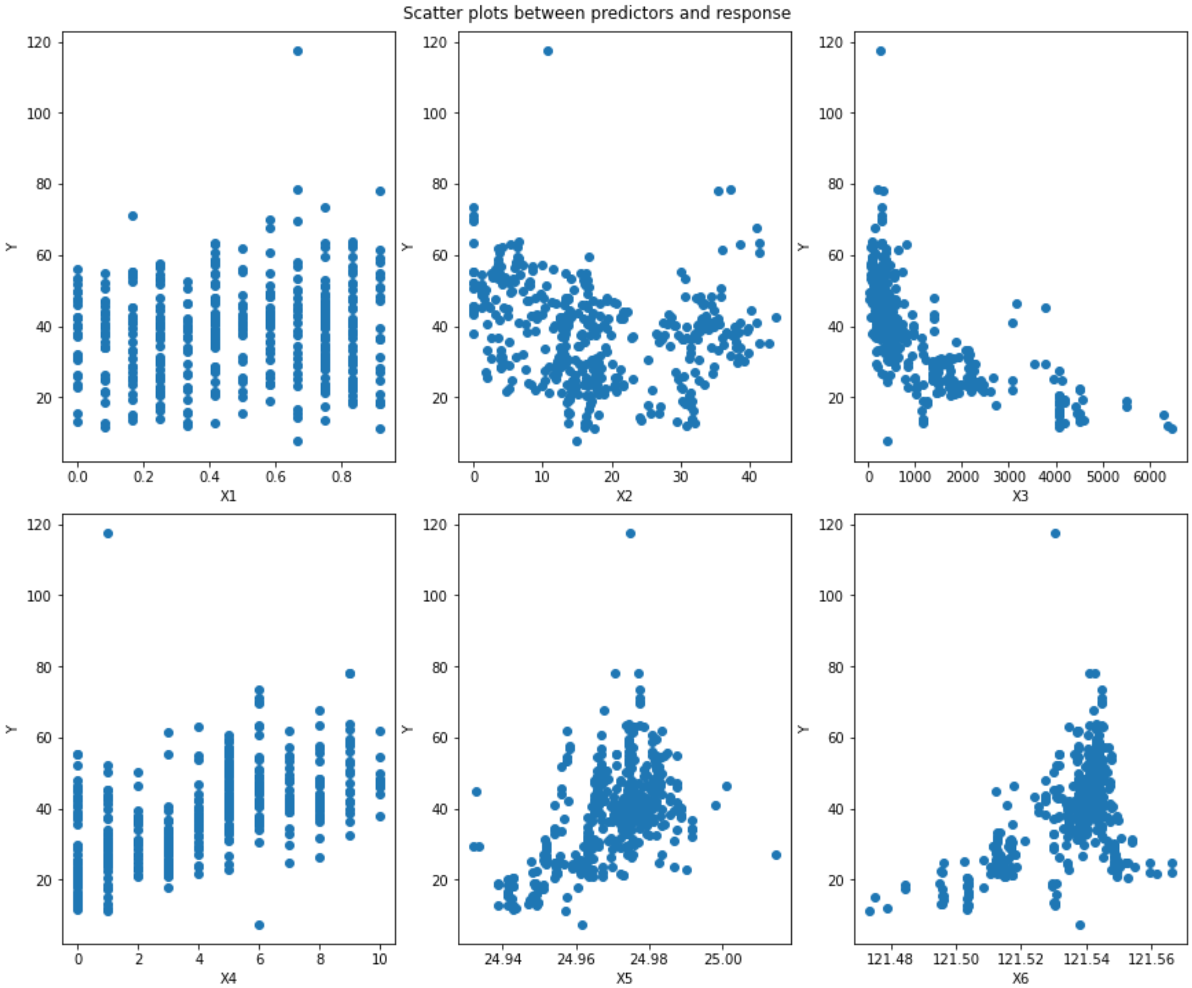
```
C:\Users\SAADMU~1\AppData\Local\Temp/ipykernel_28320/3992793283.py:4: UserWar
ning: To output multiple subplots, the figure containing the passed axes is b
eing cleared
  dfr.drop("No", axis=1).hist(ax=ax_hist, bins="auto")
```

Out[ ]: `Text(0.5, 0.98, 'Scatter plots for predictors')`



Scatter plots for predictors

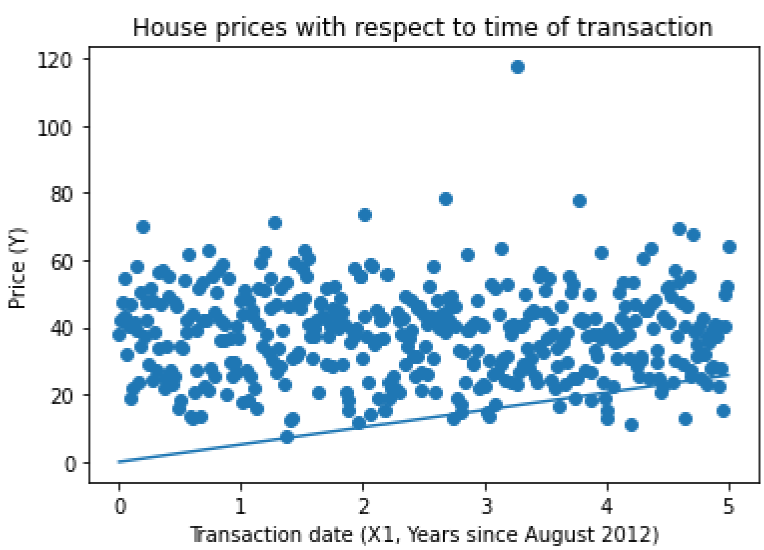



Scatter plots between predictors and response
Y
X1
X2
X3
X4
X5
X6

In [ ]:
```
# 2. Linear Regression Model
pred_names = ["X1", "X2", "X3", "X4", "X5", "X6"]
predictors = dfr[pred_names]

lin_model = LinearRegression().fit(X=predictors, y=dfr["Y"])
lin_y_preds = lin_model.predict(predictors)

def show_weights(names, weights):
    print("\n".join("{} weight: {}".format(*i) for i in list(zip(names, weight
s))))

show_weights(pred_names, lin_model.coef_)
print("\nRMSE: ", mean_squared_error(dfr["Y"], lin_y_preds, squared=False))

fig_time, ax_time = plt.subplots()
tt = np.linspace(0,5, num=414) ## Transaction times

ax_time.plot(tt, np.multiply(tt, lin_model.coef_[0]))
print(len(dfr["Y"]))
ax_time.scatter(tt, dfr["Y"])
ax_time.set_title("House prices with respect to time of transaction")
ax_time.set_ylabel("Price (Y)")
ax_time.set_xlabel("Transaction date (X1, Years since August 2012)")
```

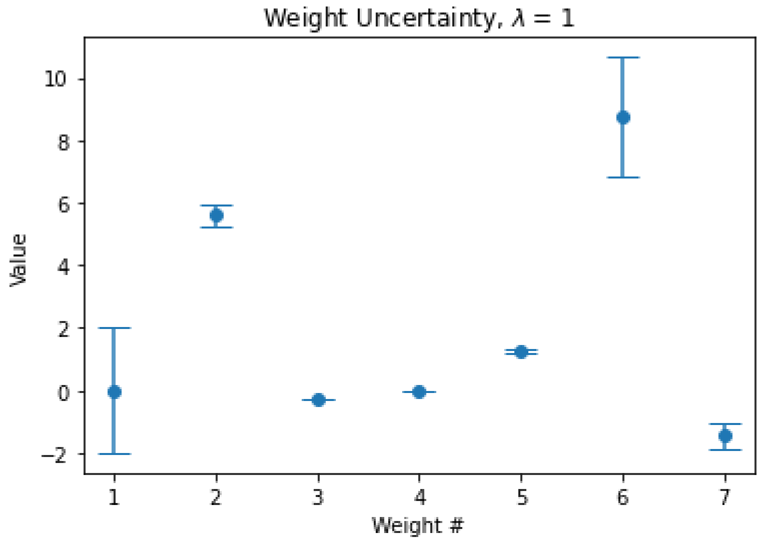
```
X1 weight: 5.149017210936201
X2 weight: -0.2696967345199864
X3 weight: -0.004487508250365868
X4 weight: 1.1333249810148465
X5 weight: 225.47014318107793
X6 weight: -12.42906116553292

RMSE:  8.782312975361108
414
```

Out[ ]:
```
Text(0.5, 0, 'Transaction date (X1, Years since August 2012)')
```

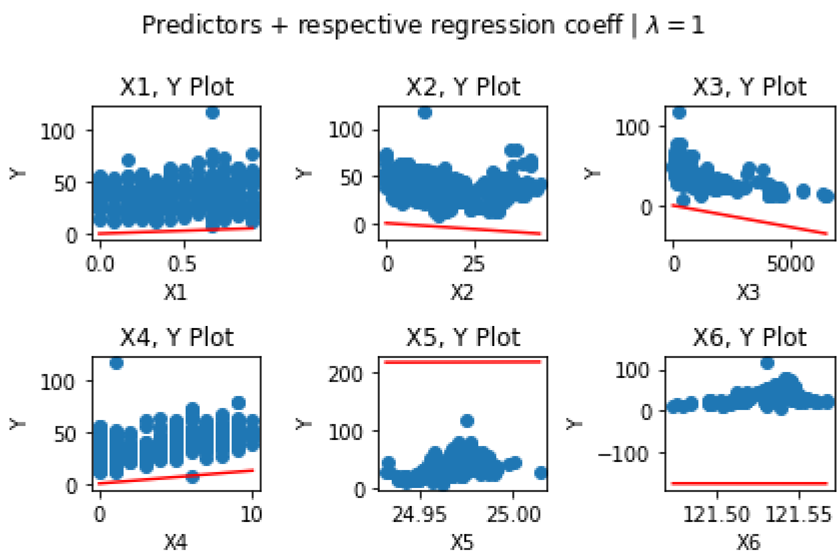


In [ ]:
```
# 3. Bayesian Linear Regression Model

# x must be a dataframe
def bayesian_reg(x, t, lambda_reg, basis=lambda v: v, basis_params=None):
    alpha = lambda_reg ** 2
    beta = lambda_reg
    def posterior(phi, t):
        # Prior here is defined as N(w|mu=0, cov=a^-1 * I)
        cov_n = np.linalg.pinv(alpha * np.eye(phi.shape[1]) + beta * phi.T.dot
(phi))
        m_n = beta * cov_n.dot(phi.T).dot(t)
        return m_n, cov_n

    def predict_dist(phi, m_n, cov_n, beta):
        cov = beta ** -1 + np.sum(phi.dot(cov_n) * phi, axis=1)
        mu = phi.dot(m_n)
        return mu, cov

    def design_mat(x, basis):
        if(basis_params == None):
            return np.concatenate([np.ones((x.shape[0], 1)), basis(x)], axis=1
)
        else:
            x.apply(lambda row: basis(row, basis_params[row.name]), axis=1)
            return np.concatenate(
                [
                    np.ones((x.shape[0], 1)),
                    x.to_numpy()
                ], axis=1)

    phi = design_mat(x, basis)
    # print("Design mat", phi)

    m_n, cov_n = posterior(phi, t)
    y, var = predict_dist(phi, m_n, cov_n, beta)

    # Frequentist approach uses penrose pseudo-inverse
    # weights = np.matmul(np.linalg.pinv(phi), t)

    # show_weights(["B0"] + pred_names, m_n)
    # print(np.diag(cov_n))
    _, ax_weights = plt.subplots()
    weight_x = np.linspace(1, len(m_n), len(m_n))
    ax_weights.errorbar(x=weight_x, y=m_n, yerr=2*np.sqrt(np.diag(cov_n)), ls=
"none", capsize=8)
    ax_weights.scatter(x=weight_x, y=m_n)
    ax_weights.set_title(f"Weight Uncertainty, $\lambda$ = {lambda_reg}")
    ax_weights.set_xlabel("Weight #")
    ax_weights.set_ylabel("Value")

    cols = min(3, x.shape[1])
    rows = x.shape[1] // cols
    fig_bay, ax_bay = plt.subplots(rows, cols)
    for i, ax in enumerate(ax_bay.ravel()):
        names = list(x.keys())
        data = x[names[i]]
```

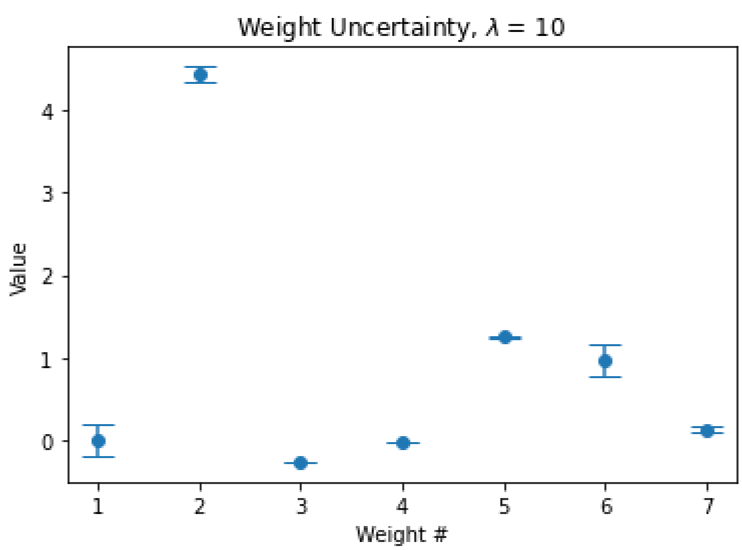
```
        ax.scatter(data, dfr["Y"])
        x_test = np.linspace(min(data), max(data), 200)
        ax.plot(x_test, (m_n[i + 1] * x_test) + m_n[0], "r")
        ax.set_title(f"X{i + 1}, Y Plot")
        ax.set_xlabel(f"X{i + 1}")
        ax.set_ylabel("Y")
    fig_bay.suptitle(f"Predictors + respective regression coeff | $\lambda={lambda_reg}$")
    fig_bay.tight_layout()



bayesian_reg(predictors, dfr["Y"], lambda_reg=1)
bayesian_reg(predictors, dfr["Y"], lambda_reg=10)
bayesian_reg(predictors, dfr["Y"], lambda_reg=100)
```



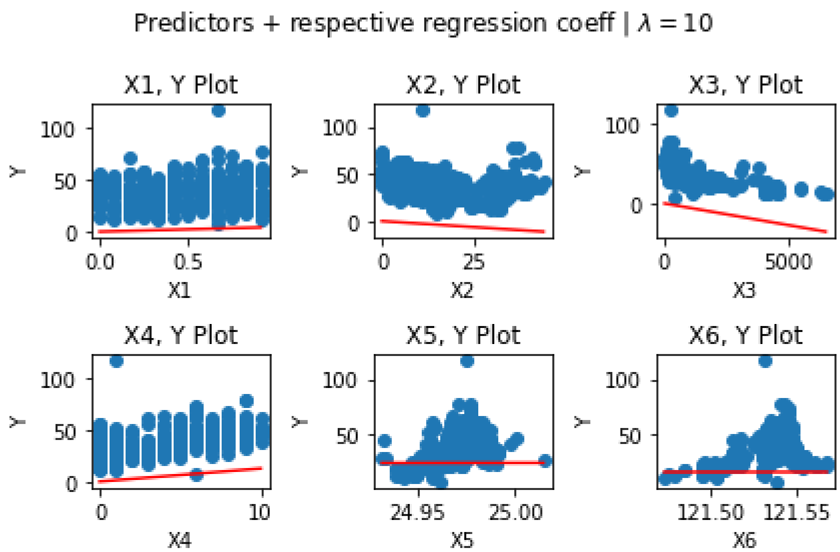
Weight Uncertainty, λ = 1
Value
Weight #

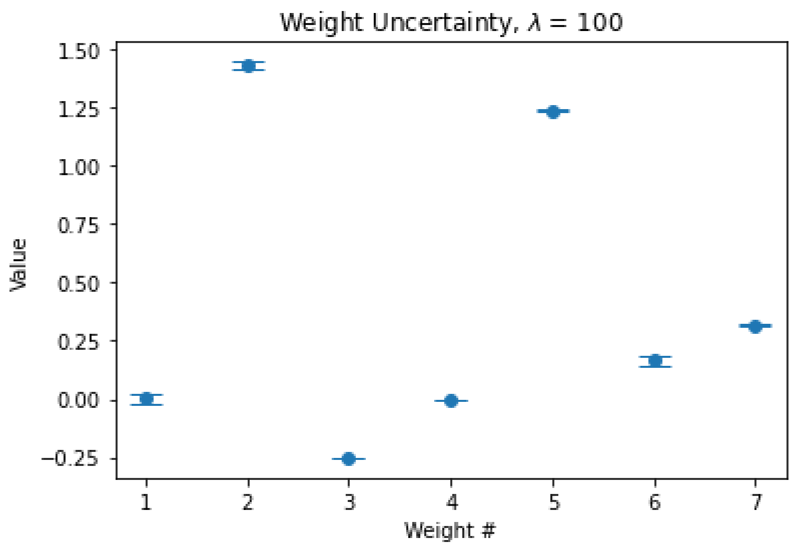


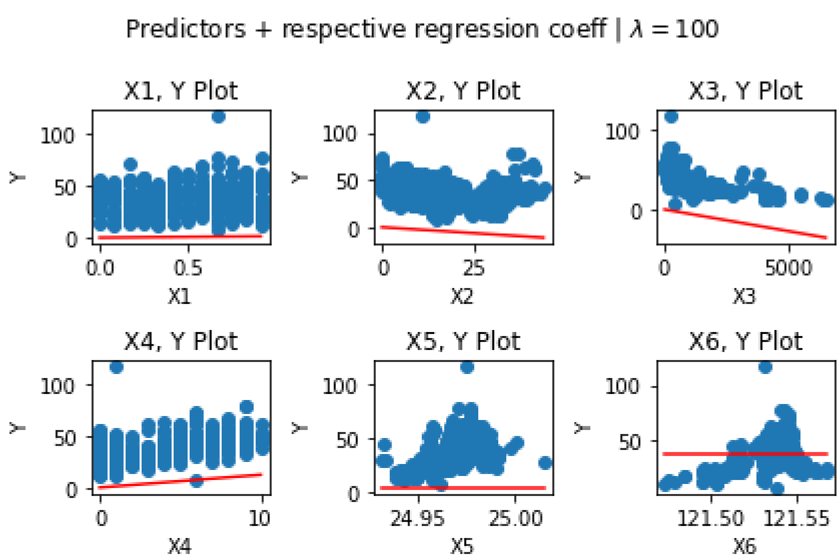
Predictors + respective regression coeff | $\lambda = 1$
X1, Y Plot
X2, Y Plot
X3, Y Plot
X4, Y Plot
X5, Y Plot
X6, Y Plot
X1
X2
X3
X4
X5
X6
Y



Weight Uncertainty, λ = 10
Value
Weight #


Predictors + respective regression coeff | $\lambda = 10$
X1, Y Plot
X2, Y Plot
X3, Y Plot
X4, Y Plot
X5, Y Plot
X6, Y Plot
Y
X1
X2
X3
X4
X5
X6
0
50
100
0.0
0.5
25
5000
10
24.95
25.00
121.50
121.55


Weight Uncertainty, λ = 100
Value
Weight #
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
1
2
3
4
5
6
7


Predictors + respective regression coeff | $\lambda = 100$
X1, Y Plot
X2, Y Plot
X3, Y Plot
X4, Y Plot
X5, Y Plot
X6, Y Plot
Y
X1
X2
X3
X4
X5
X6
0
50
100
0.0
0.5
25
5000
10
24.95
25.00
121.50
121.55

In [ ]:

```python
# 3. Bayesian Linear Regression, SKLearn
# I attempted to perform this problem with a custom made solution, but the res
ults were poor
def bay_ridge(x, t, lambda_reg, basis=lambda v: v, basis_params=None):
    model = BayesianRidge(lambda_1=lambda_reg)
    design_mat = x.copy()

    if basis_params is not None:
        for c in x.columns:
            for i in basis_params:
                design_mat[f"{c}.mu{i}"] = basis(x[c], i)
    # print(design_mat.shape)
    model.fit(X=design_mat, y=t)
    predicted, std = model.predict(X=design_mat, return_std=True)
    print("Predicted vals", predicted)
    print("Uncertainties", std)
    weights = model.coef_
    weight_std = np.diag(model.sigma_)
    intercept = model.intercept_
    print("Intercept", intercept)

    _, ax_weights = plt.subplots(constrained_layout=True)
    weight_x = np.linspace(1, len(weights), len(weights))
    ax_weights.errorbar(x=weight_x, y=weights, yerr=2*weight_std, ls="none", c
apsize=8)
    ax_weights.scatter(x=weight_x, y=weights)
    ax_weights.set_title(f"Weight Uncertainty, $\lambda$ = {lambda_reg}")
    ax_weights.set_xlabel("Weight #")
    ax_weights.set_ylabel("Value")

    cols = min(3, design_mat.shape[1])
    rows = design_mat.shape[1] // cols
    fig_bay, ax_bay = plt.subplots(rows, cols, constrained_layout=True)
    for i, ax in enumerate(ax_bay.ravel()):
        names = list(design_mat.keys())
        data = design_mat[names[i]]
        ax.scatter(data, dfr["Y"])
        x_test = np.linspace(min(data), max(data), 200)
        ax.plot(x_test, (weights[i] * x_test) + intercept, "r")
        ax.set_title(f"{names[i]}, Y Plot")
        ax.set_xlabel(names[i])
        ax.set_ylabel("Y")
    fig_bay.suptitle(f"Predictors + respective regression coeff | $\lambda={la
mbda_reg}$")
    # fig_bay.tight_layout()
    del design_mat
    return model


bay_1 = bay_ridge(x=predictors, t=dfr["Y"], lambda_reg=1)
bay_2 = bay_ridge(x=predictors, t=dfr["Y"], lambda_reg=10)
bay_3 = bay_ridge(x=predictors, t=dfr["Y"], lambda_reg=100)
```

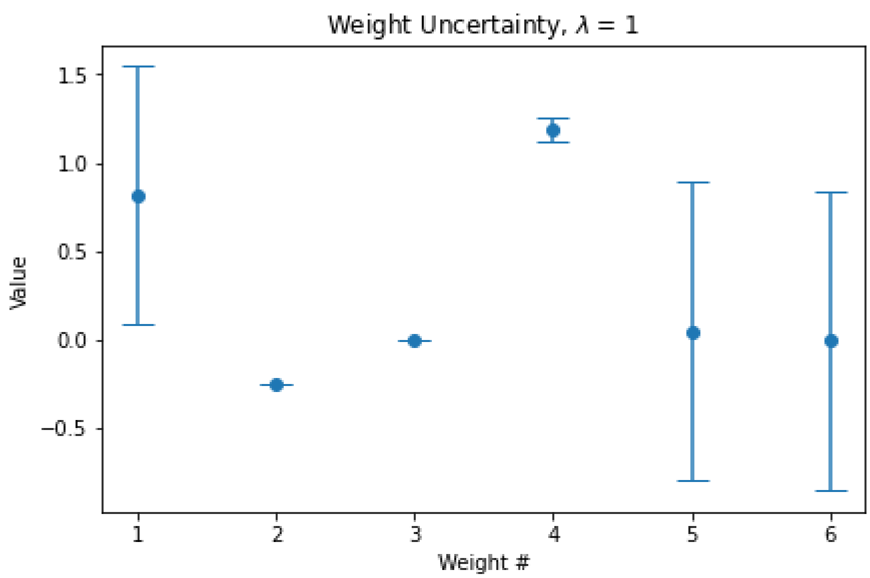
```
Predicted vals [46.7630763  47.4763383  43.41025225 43.3421758  45.82610157 3
2.89297689
 39.38533815 44.22960541  6.53244148 32.97500145 33.74046204 52.3202548
 43.32659262 29.11549027 38.83715658 34.14562837 49.14619003 38.05136342
 47.00708079 50.97572461 33.59735344 47.91659968 33.33963169 47.74444785
 35.61204881 30.29982552 46.20293543 45.44622948 40.70256166 45.16778658
 12.3015666  39.8711181  32.2296225  44.8448585  46.55489837 17.75111375
 31.33879059 34.21990148 46.31269166 43.85573832 17.39993207 17.09279735
 37.80002021 38.90596154 45.0350606  41.13296824 46.46062357 34.94075528
 12.1817702  11.98299124 40.25769979 27.11055479 31.46724917 43.38762992
 43.81270032 28.93480615 42.80865104 50.49517928 12.43840919 44.25349996
 32.69475097 48.20819292 30.56361595 45.06009913 33.81370007 42.55405151
 49.24240696 47.07866001 40.72111998 42.99787221 52.44936864 34.63245055
 42.92481226 17.48600799 51.43845761 34.41709176 35.07581819 29.61796968
 33.12331044 32.27805261 42.68847776 40.75917821 47.67912768 25.58747685
 45.91473174 52.04254392 35.97092987 17.13583527 33.35016768 16.19867182
 42.97739293 33.59687568 29.26964264 29.03210602 38.14137807 46.72731737
 52.3632928  36.53951421 44.00989069 52.3632928  38.87010715 40.37362947
 49.28544488 49.20600064 39.92985139 48.80580762 47.65079832 34.30597731
 35.0120099  34.57138422 46.83843173 42.80194845 33.20712143 44.95176331
 44.2088341  36.63767057  1.41971702 16.76012713 32.29640693 44.02789007
 43.45570645 43.65289673 38.20943611 42.75832241 47.65837201 49.35352141
 34.1338255  46.50420101 39.60044583 41.25186085 42.76334554 34.50330777
 40.17855824 46.22691335 40.89431776 33.04803936 44.15555909 45.79997922
 35.85191442 43.14740172 43.92381477 37.35085752 41.20967706 43.65289673
 34.17686348 46.28356027 42.21371058 49.21763834 18.77862255 42.93424638
 42.04677211 51.15831924 34.35605438 47.05631113 17.06775882 17.48600799
 29.22048636 39.84132271 44.30971145 39.5150186  51.71974311 39.12110009
 16.68041732 47.07866001 42.28178703 34.27656465 49.28234301 44.38911712
 39.68007034 31.98403104 12.59674811 46.4181251  52.44936864 35.69204141
 48.62145898 36.89412517 14.46161161 44.92826217 38.97535636 38.76202668
 11.76101769 49.05278485 32.12464726 16.95708085 25.20463889 30.91339928
 29.79012152 24.14450251 43.10497339 16.7414547  41.0908021  38.49082332
 40.60315696 44.95452048 19.12734687 43.28823553 36.18519252 43.82818725
 42.43407365 46.21852651 33.90681513 43.32417641 33.7843496  43.59751065
 32.27805261 35.82936341 47.8548296  33.74635681 34.00471173 43.11664618
 46.32063634 43.18162235 31.67149489 52.14106391 33.06107728 46.27694195
 32.09532007 41.42153483 43.58482028 42.46189751 44.05074028 33.68999066
 45.73786154 33.59687568 40.39885651 49.42159786 16.67381465 43.29000063
 23.21967138 29.7461256  34.50330777 16.65318903 13.30443868 42.61043831
 34.10012295 46.92893351 52.47562661 36.26324049 36.24524103 34.67613584
 33.63473076 47.34131354 34.84765307 43.9361104  37.22174369 44.29499717
 44.63837892 32.43307687 33.09107728  5.01410864 39.71909963 29.00941169
 52.01195007 28.30954266 47.0026636   6.5144421  38.82392923 38.04261601
 49.28234301 34.43590756 35.90433057 30.39609491 43.72662449 34.46026985
 40.9889955  39.60109443 32.863887   37.87813121 43.24793609 34.39286964
 40.80055047 45.93526031 43.19043964 40.7249722  42.81879027 50.97572461
 42.03942691 29.91923535 49.20600064 38.07573733 49.17246743 51.07224332
 35.98892934 26.79892445 45.80361737 35.62244753 52.0800266  42.01438838
 45.09192515 44.56777745 31.35537809 50.45214128 42.8711577  46.20031271
 42.50449383 30.49686495 38.51289926 36.17363841 16.89604347 46.46241708
 46.76860322 31.16636477 30.59328519 31.82633623 33.3204298  43.78766179
 40.49340983 23.64830183 44.00989069 29.04833461 33.07379904 40.013604
 43.97315692 46.9925841  46.63899577 32.24106163 47.23723824 37.5348719
 51.70372434 39.40780603 16.58093624 38.54380303 40.3634671  42.65295789
 32.94380899 41.13296824 50.8214834  42.2702979  34.58263202 16.76012713
 29.08528229 12.24052921 32.56486624 46.64124145 36.1810501  42.33498405
```

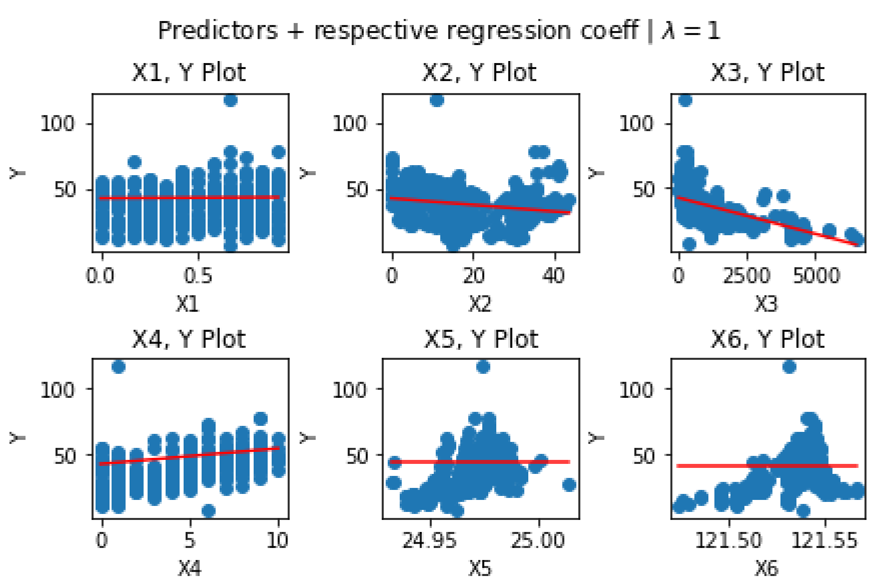
```
 34.05127539 38.07408568 44.7396106  47.39704417 33.80339119 38.4047474
 51.92587424 41.42872504 18.0817261  42.14563413 33.35618902  4.76833808
 47.83885081 46.64124145 43.344592   33.77641056 27.42697265 34.47826924
 34.37405384 46.50420101 40.74541224 52.58715575 49.35352141 29.09077254
 46.32006036 41.08279281 38.88748519 47.02540877 39.84947406 31.25074413
 30.88046406 31.78736314 38.4707826  29.55693238 43.59047151 47.02573565
 42.39103573 43.18162235 46.06632985 31.79414411 32.95533291 51.95997131
 40.61248748 49.21426648 44.51770039 51.72967095 19.78739197 37.82805867
 16.65537878 50.27248277 42.28178703 31.48728378 33.50747627 42.22483112
 44.22720152 32.79777997 37.48231231 38.59717985 13.93761133 39.97443359
 29.97641962 38.79411866 32.95533291 40.44170601 40.19655762 33.10816656
 40.27739119 45.22015328 43.94181416 43.80834876 49.34999161 28.78246781
 30.6473254  17.37489354 51.95091277 45.08242212 46.77035528 52.40633072]
Uncertainties 0      81.305754
1      81.298201
2      81.292031
3      81.291730
4      81.285339
         ...
409    81.296420
410    81.294246
411    81.292503
412    81.280000
413    81.295548
Length: 414, dtype: float64
Intercept 42.82402470182194
Predicted vals [44.77951529 43.2731756  41.53551979 41.53551941 42.70517454 3
0.55273881
 41.1109073  43.40206546  7.82149127 33.21806008 42.5966795  44.74877811
 42.01060222 28.54277946 37.43020904 41.4116404  43.37110024 42.97173312
 42.85476413 45.20674671 29.86674167 43.46233135 36.09961695 43.46244364
 42.08150247 35.22573628 42.75138041 43.4807617  41.56445652 42.29139535
 14.5806866  40.11855666 42.02814199 43.15763831 43.96352994 17.58223237
 32.18498238 36.10038634 41.42969103 43.39143036 17.56463057 17.66878279
 41.81860021 41.86460523 41.73259739 42.02733921 42.20092179 40.99264781
 13.99513179 14.64330062 41.86970099 33.38280561 35.56065857 42.01051762
 43.39145844 37.45329517 42.82881237 44.98083537 14.64304835 43.07404756
 32.21192802 43.59660984 30.54409831 41.73262584 38.58010219 44.51241482
 44.04779181 44.6501172  42.19645459 41.53574399 44.74869389 40.99275934
 42.46620386 17.56457443 42.78000473 36.1003025  41.15838767 30.4801622
 41.59323053 35.72620002 41.73000774 42.78488571 44.33482644 26.92232249
 42.75207478 43.05827067 35.45187557 17.66875471 35.78596612 18.47493809
 43.5000093  35.81596886 28.54272369 37.55063815 44.21332114 44.65025681
 44.74875003 41.15342366 43.3913746  44.74875003 38.79153256 44.20382742
 44.04776374 43.94725041 42.68457905 43.37109834 44.07101811 36.1003302
 41.32234623 30.7419412  44.65022911 44.01802431 31.04635202 42.68412739
 44.37827033 40.33517809  1.80025115 16.77923772 34.57420907 43.39131807
 42.010518   42.01043417 42.53154846 44.10317796 43.46249979 44.04776412
 39.8754909  42.7511839  44.50342697 43.87888061 41.71126512 30.74194081
 42.07124358 42.81864704 44.08362383 38.48790957 42.70335313 43.18968174
 38.94511889 42.01062953 43.39143075 34.7426743  40.99827125 42.01043417
 36.10041441 42.29281727 44.10317491 42.02960377 19.61645758 44.13451506
 44.19358868 42.72453669 36.1003871  42.8028648  17.66875433 17.56457443
 36.74470436 39.80534081 42.70329737 39.80550886 45.02730355 41.16313864
 17.66900699 44.6501172  44.10317529 36.94141329 43.371101   43.11066507
 41.85063613 31.99862728 14.52676908 42.75124004 44.74869389 42.61774126
 42.4231207  42.14214597 14.21576675 44.12407891 37.5691695  42.37353297
```

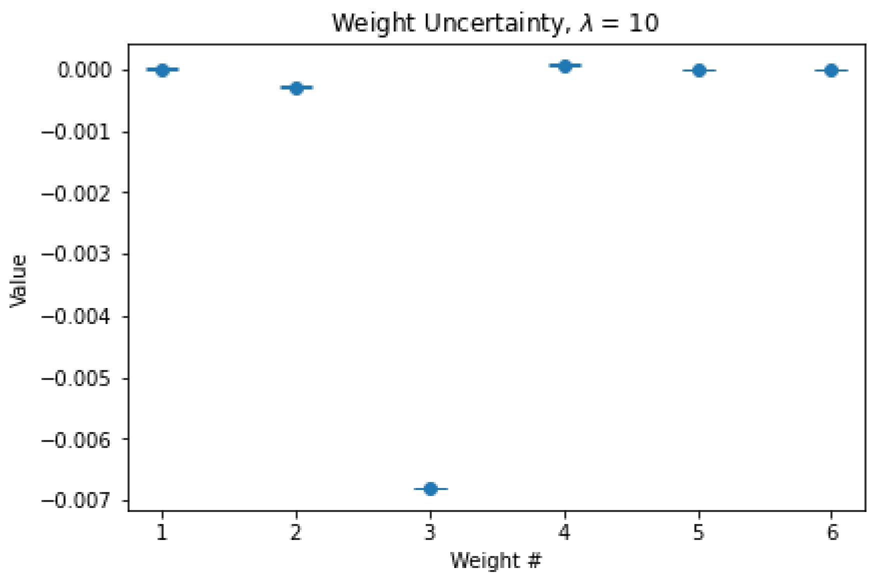
```
 15.05993911 43.98833609 30.73923773 17.56366544 27.55128966 35.50156165
 30.48004991 24.40183563 44.06246099 17.66892239 41.15774002 40.25451658
 44.96177556 42.4933472  19.67600958 42.22196285 40.5389075  44.49370411
 44.28348385 42.28574405 38.58013103 41.53557594 40.9786189  43.3915988
 35.72620002 35.49026703 42.77562442 40.82626257 36.1005267  44.16058739
 42.70512068 43.50001044 33.06495624 44.74880542 33.21800394 42.74803696
 41.33115309 42.82089784 42.01043379 41.75050343 44.08588437 35.8159977
 42.42216796 35.81596886 43.14375363 44.0477645  17.56380542 43.55226671
 23.76663761 37.48223481 30.74194081 17.61345801 15.30854452 43.08511177
 30.26939555 43.65600587 42.81985848 40.3716763  40.37173283 39.65608741
 33.70036219 43.65578167 31.21909226 43.96680208 34.74275852 41.00849427
 42.70708911 38.17501145 38.4878815   2.41768142 42.46307257 37.46131741
 44.74888964 33.53846859 43.11890991  7.8215478  43.05176151 42.33662511
 43.371101   39.65619932 35.24825087 29.78078226 43.39151458 30.74196889
 41.99546217 39.80545272 33.21808777 42.06893733 42.70170588 39.65622739
 43.642886   42.28588403 42.01060146 44.20368782 42.67543227 45.20674671
 42.22085164 30.47996569 43.94725041 34.78136385 44.1104423  42.72459283
 35.45181904 31.8823092  42.75210247 40.46617069 44.74889002 42.22082319
 43.15736487 43.39208556 42.01606211 44.98086345 42.66684649 42.75278421
 43.07397546 30.51853041 37.5693368  41.49490642 17.56375004 44.52825446
 44.30149827 42.21045962 29.78069842 42.3606976  34.28724996 43.39142998
 44.20435868 24.38976465 43.3913746  36.74481665 34.16960471 41.697879
 43.18738388 44.65017335 41.42952297 33.40093037 43.65589435 40.23546937
 42.77992127 43.49955339 16.77926503 38.79170061 44.08559939 44.0157708
 33.69996193 42.02733921 44.98066732 40.20786155 35.16257831 16.77923772
 37.47946245 14.5807712  41.14826275 44.65031295 38.45715128 43.05959434
 32.64924948 41.26566054 43.59941112 43.11874223 43.51165877 40.25457272
 44.74894578 41.52112721 24.34689816 44.10317453 33.69973773  1.17938447
 43.59680597 44.65031295 42.0105457  30.51810891 27.1451316  30.74191236
 36.10033058 42.7511839  43.92314335 43.05827372 44.04776412 28.95972922
 44.76275537 43.43976977 38.77359749 44.60873539 41.17424303 29.96153597
 33.09391171 32.91142622 42.9715939  30.4802468  43.39151382 43.23435931
 44.28351192 43.50001044 42.70506263 37.47414734 33.66825177 45.02719164
 41.3525126  43.37110062 43.39202865 44.46166773 21.32590182 41.91140782
 17.66897853 44.79698142 44.10317529 31.03687731 30.02815346 44.52159066
 42.78429977 32.15505372 42.33201468 38.7736525  17.1898304  41.86984097
 39.09975839 37.43023711 33.66825177 44.2038278  42.07118706 30.55259844
 44.08565553 44.2549759  43.39137422 44.46775122 42.8317043  28.95984075
 30.54463639 17.56460212 44.74897423 42.69866211 44.65022873 44.74872196]
Uncertainties 0      10.072060
1      10.072607
2      10.074169
3      10.074169
4      10.072980
         ...
409    10.190767
410    10.071951
411    10.073022
412    10.071968
413    10.071952
Length: 414, dtype: float64
Intercept 45.36588890669055
Predicted vals [38.31417203 38.24005941 38.15468069 38.15468069 38.21199685 3
7.61541057
 38.13410421 38.2464061  36.49975339 37.7464036  38.20706868 38.31233209
 38.17800093 37.51690218 37.95313357 38.14889833 38.24462234 38.22525984
 38.21948666 38.33475945 37.58169788 38.24923472 37.88783686 38.24923511
```

38.18182998	37.84512067	38.21424067	38.25014403	38.15618082	38.19170972
36.83152114	38.08532148	38.17919686	38.23435674	38.27390464	36.9787251
37.69564646	37.88783953	38.14934995	38.24583367	36.97785716	36.98301192
38.16887449	38.17110828	38.16422138	38.1791204	38.18744597	38.12832777
36.80275149	36.8346379	38.17119916	37.75467152	37.86159694	38.17800064
38.24583377	37.95451881	38.2184294	38.32369433	36.83463702	38.23021424
37.6969212	38.25576241	37.61511745	38.16422148	38.0096545	38.30116963
38.2778573	38.30752838	38.18734823	38.15468147	38.3123318	38.12832816
38.20061295	36.97785697	38.2156796	37.88783923	38.13646471	37.6120214
38.15784603	37.86954693	38.16421241	38.21624232	38.29210467	37.43741289
38.21442474	38.22925558	37.85586876	36.98301182	37.87236586	37.02267031
38.25096553	37.87384136	37.51690199	37.95930041	38.28650485	38.30752887
38.31233199	38.13612373	38.24583348	38.31233199	38.02002307	38.28568412
38.2778572	38.27290833	38.21134254	38.24462234	38.27920216	37.88783933
38.14445867	37.62465895	38.30752877	38.27681226	37.63968052	38.21108781
38.29446067	38.09585386	36.20413256	36.93929859	37.81307929	38.24583328
38.17800064	38.17800034	38.2038196	38.28058062	38.24923531	38.2778572
38.07351366	38.21423999	38.30074977	38.2700486	38.16351193	37.62465895
38.18115462	38.21772791	38.28003721	38.00514661	38.21199054	38.23589219
38.02747752	38.17800103	38.24583367	37.82108779	38.1283891	38.17800034
37.88783962	38.19171465	38.28058061	38.17879798	37.07862709	38.28254943
38.28546406	38.21293449	37.88783953	38.21680943	36.98301182	36.97785697
37.91971766	38.06977749	38.21199035	38.06977808	38.3259728	38.13648116
36.9830127	38.30752838	38.28058061	37.92915156	38.24462234	38.23219674
38.17033698	37.68641839	36.82884944	38.21424018	38.3123318	38.20817848
38.19813456	38.18468805	36.81344987	38.28200944	37.95995448	38.19584331
36.85506302	38.27507019	37.62464959	36.97785382	37.46815503	37.85868669
37.61202101	37.31346618	38.27901061	36.9830124	38.13641078	38.09179772
38.3232818	38.20165816	37.0815352	38.18840608	38.10588446	38.30017707
38.28985409	38.19156771	38.0096546	38.15468088	38.12767033	38.24583426
37.86954693	37.85789061	38.21566443	38.12019301	37.88784001	38.28382811
38.21199665	38.25096553	37.73888601	38.31233218	37.7464034	38.21427681
38.14497705	38.21797398	38.17800034	38.16543454	38.28018877	37.87384145
38.19842061	37.87384136	38.23390877	38.2778572	36.9778543	38.25393165
37.28232023	37.95593034	37.62465895	36.98028812	36.86726788	38.23108867
37.60150863	38.2587744	38.2175946	38.09755278	38.09755297	38.06248735
37.76999539	38.25877362	37.64805869	38.2742882	37.82108808	38.12873129
38.21223586	37.98982126	38.00514651	36.23427761	38.20054918	37.95491264
38.31233248	37.76229927	38.23225843	36.49975358	38.22914857	38.19407415
38.24462234	38.06248774	37.84606054	37.5776297	38.24583396	37.62465904
38.17750632	38.06977788	37.74640369	38.18114664	38.21198483	38.06248784
38.25812013	38.1915682	38.17800093	38.28568363	38.21082299	38.33475945
38.18840223	37.61202072	38.27290833	37.82295476	38.28094993	38.21293469
37.85586857	37.68103083	38.21442483	38.10240535	38.31233248	38.18840213
38.23434454	38.24583594	38.17860401	38.32369443	38.21027739	38.2144272
38.23037783	37.61387514	37.95995507	38.15297027	36.97785411	38.301852
38.29033812	38.18815185	37.57762941	38.19553146	37.79892234	38.24583367
38.28573213	37.31289177	38.24583348	37.91971805	37.79310269	38.16275848
38.2360538	38.30752858	38.14934937	37.75535463	38.25877402	38.09089385
38.21567931	38.25138082	36.93929869	38.02002366	38.27988232	38.27664351
37.76999401	38.1791204	38.32369375	38.08937833	37.84184315	36.93929859
37.95580732	36.83152143	38.13602123	38.30752907	38.0037663	38.22968347
37.71831485	38.14166592	38.25623163	38.23225784	38.25197648	38.09179791
38.31233267	38.15422787	37.31110263	38.28058061	37.76999323	36.17347593
38.25576309	38.30752907	38.17800073	37.61367014	37.44826204	37.62465885
37.88783933	38.21423999	38.27187287	38.22925559	38.2778572	37.53719132
38.31336094	38.24852547	38.01913738	38.30579142	38.13722388	37.5865152

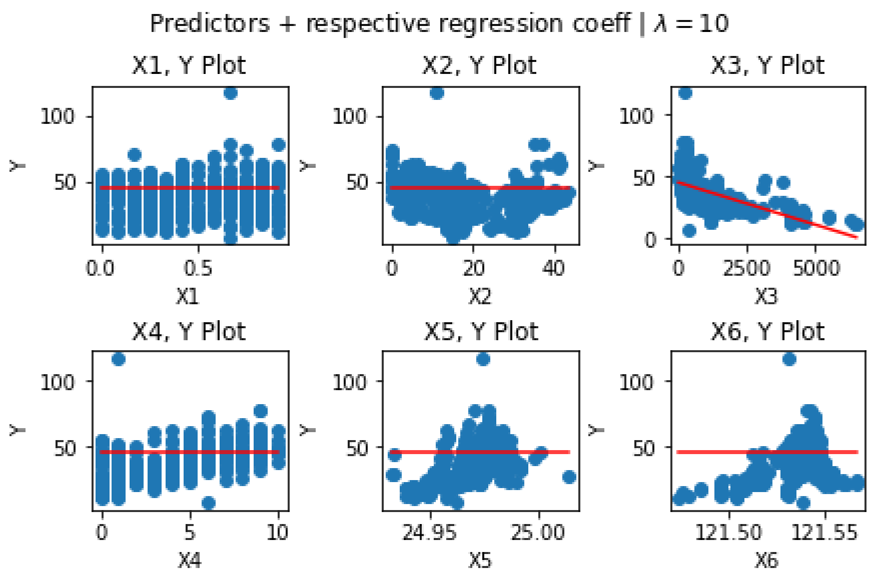
```
 37.74026642 37.73131486 38.22525936 37.61202169 38.24583396 38.23796515
 38.28985419 38.25096553 38.21199646 37.95541261 37.76846691 38.32597241
 38.14599907 38.24462234 38.24583574 38.29825857 37.16255043 38.17334299
 36.9830126  38.31485162 38.28058061 37.6392972  37.58969605 38.30163054
 38.2162308  37.69406949 38.19416327 38.01913758 36.9597002  38.17119965
 38.03541859 37.95313367 37.76846691 38.28568412 38.18115443 37.61541008
 38.27988251 38.2884086  38.24583347 38.29876077 38.21816271 37.53719171
 37.61516064 36.97785706 38.31233277 38.21185047 38.30752877 38.31233189]
Uncertainties 0      13.015519
1      13.015558
2      13.015659
3      13.015659
4      13.015585
         ...
409    13.023101
410    13.015519
411    13.015585
412    13.015520
413    13.015519
Length: 414, dtype: float64
Intercept 38.34257315585334
```



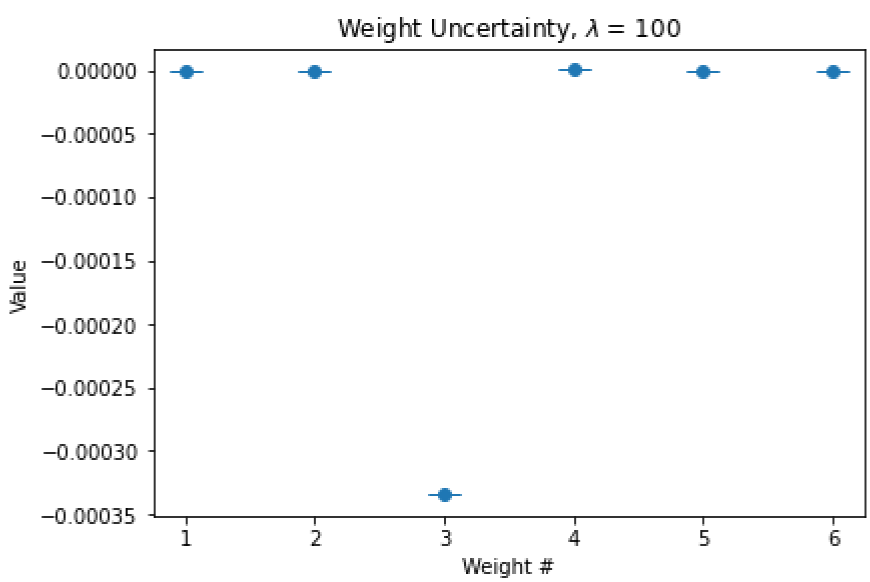




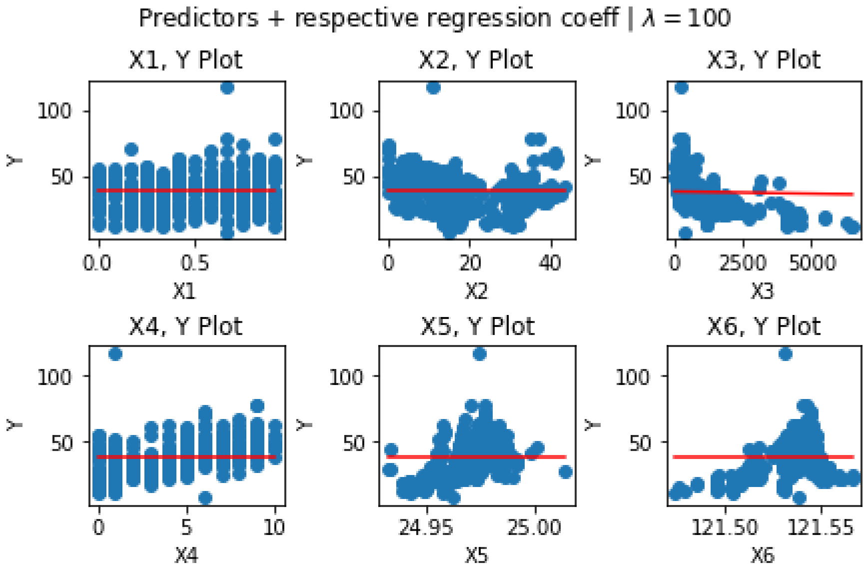
Weight Uncertainty, λ = 10
Value
Weight #
0.000
−0.001
−0.002
−0.003
−0.004
−0.005
−0.006
−0.007
1
2
3
4
5
6



Predictors + respective regression coeff | $\lambda = 10$
X1, Y Plot
X2, Y Plot
X3, Y Plot
X4, Y Plot
X5, Y Plot
X6, Y Plot
Y
X1
X2
X3
X4
X5
X6



Weight Uncertainty, λ = 100
Value
Weight #
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
−0.00030
−0.00035
1
2
3
4
5
6


Predictors + respective regression coeff | $\lambda = 100$
X1, Y Plot
X2, Y Plot
X3, Y Plot
X4, Y Plot
X5, Y Plot
X6, Y Plot
Y
X1
X2
X3
X4
X5
X6
100
50
0.0
0.5
0
20
40
2500
5000
5
10
24.95
25.00
121.50
121.55

In [ ]:

```
# 4. Standardization + Gaussian Basis
standardized_pred = preprocessing.StandardScaler().fit_transform(dfr[["X2", "X
3"]])
standardized_pred = pd.DataFrame(standardized_pred, columns=["X2", "X3"])
gauss = lambda x_i, mu: np.exp(-0.5 * (x_i - mu) ** 2)

mu = [-3, -2, -1, 1, 2, 3]

stan_1 = bay_ridge(standardized_pred, dfr["Y"], lambda_reg=1, basis=gauss, bas
is_params=mu)
stan_2 = bay_ridge(standardized_pred, dfr["Y"], lambda_reg=10, basis=gauss, ba
sis_params=mu)
stan_3 = bay_ridge(standardized_pred, dfr["Y"], lambda_reg=100, basis=gauss, b
asis_params=mu)
```

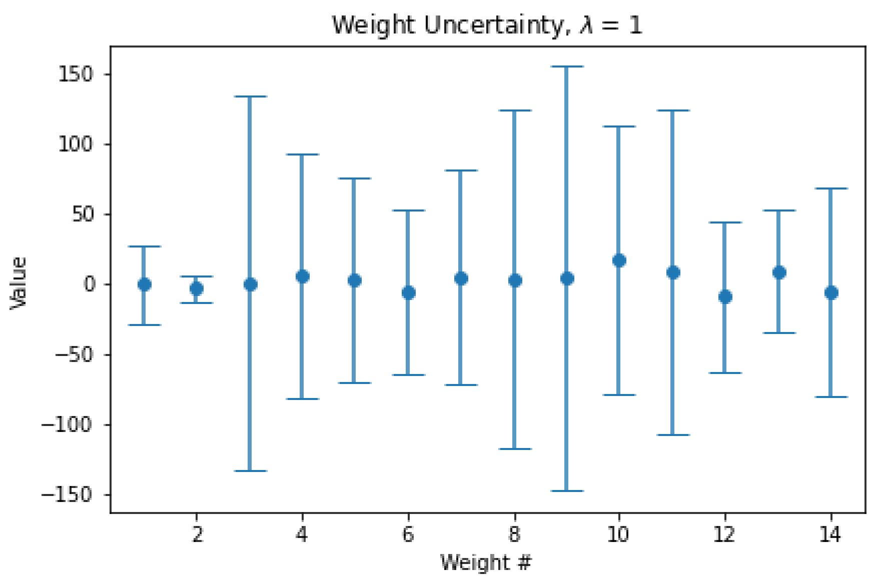
```
Predicted vals [44.58188796 41.89539606 41.43008724 41.43008724 48.80107561 2
8.52232292
 37.40349029 41.83998776 12.34728656 24.13436823 40.87340081 52.71758074
 42.68101892 20.40291666 32.96162858 38.6295193  52.06745862 42.10151957
 42.2685782  55.70701719 29.52743573 47.48771547 29.75172232 47.72982704
 41.98631784 24.416511   49.71434533 47.59001164 38.18840998 46.81316275
 16.02062349 33.51852052 41.07057757 43.1741997  45.61299943 22.74713081
 25.00935099 31.41020925 46.74251551 43.87002193 22.92158826 21.08632962
 39.73098359 39.04295777 47.57088938 40.4401105  38.61286272 37.79277308
 15.76385402 16.35216002 37.86850607 22.63279954 25.70076856 42.49584454
 43.92737846 28.73907505 40.86651845 54.50865508 16.56110343 44.89225925
 27.30465161 50.66413504 22.53464619 47.6073213  32.65209086 47.6494413
 53.32263127 51.30555333 38.31681639 41.92453271 52.56242884 37.6034577
 39.60769706 22.79883976 48.07355751 31.22460054 38.16035006 20.92645828
 40.23768073 27.30035216 42.79238589 39.75485095 47.7677001  19.06778413
 43.07341561 51.36364952 35.61594302 21.0308767  32.78784389 18.86084364
 52.35741922 32.71763383 20.32266527 28.98783163 47.33241098 51.58945086
 52.66618836 35.57477108 43.75606925 52.66618836 33.0766136  47.78807307
 53.29408449 53.35547931 40.17696157 52.06745862 44.82879981 31.28648591
 36.65316323 30.11600576 51.53315056 43.40688127 27.17629828 43.09968506
 43.25860079 34.86925218 10.13658508 22.46859183 23.29531071 43.64317659
 42.49584454 42.31117299 39.40119041 53.70273106 47.85007858 53.29408449
 36.65616317 49.43371479 48.13370579 45.51594341 37.03192791 30.11600576
 37.499198   41.59920903 43.59395237 31.75849662 45.23000656 44.96778006
 37.87019265 42.74282541 43.87002193 33.29730261 36.65682708 42.31117299
 31.47203489 49.04171497 53.70273106 48.04865945 22.13888331 44.36506189
 44.91726326 48.89036913 31.41020925 48.28261191 21.0308767  22.79883976
 27.16318426 35.96105707 45.10674215 36.31008347 54.60882769 37.29354182
 21.53990942 51.30555333 53.70273106 31.77657901 52.06745862 39.94468419
 37.04644352 28.68497197 16.19659368 49.51611083 52.56242884 43.92126247
 48.48052653 38.14281014 20.50573782 43.52016302 33.27981386 42.88468356
 16.36294998 47.98870362 24.80200738 20.91952602 23.66486211 25.37273398
 20.77031668 27.03114282 44.159489   21.36784674 37.87468328 38.66641043
 49.99436243 45.78832764 22.85502063 41.81928082 34.58520081 44.93039243
 44.29129719 40.29551554 32.70564174 41.55349171 38.9164582  44.21768648
 27.30035216 29.96969026 39.74989862 38.67485201 31.71895475 44.37696052
 48.7071959  52.35741922 24.13840973 52.76864214 24.031192   40.85073083
 39.46061975 39.26936046 42.31117299 37.05779158 45.35193527 32.77614925
 38.87942616 32.71763383 41.9690936  53.29408449 21.1982197  42.01395899
 25.26788128 28.56760076 30.11600576 21.39540473 16.5480815  44.29218099
 27.90202513 46.44480164 49.66997639 39.04440971 39.1680529  34.55455317
 28.73694651 45.95173402 31.13891302 43.09214576 33.43638372 43.72482093
 42.21581915 29.99726399 31.71078178 12.01489094 42.66288222 28.75383728
 52.91979567 22.53587331 51.21603954 12.28049374 44.049231   40.88309743
 52.06745862 34.7623806  27.12062217 22.78884748 44.04281731 30.15828368
 38.58555224 36.19285644 24.18651289 39.63790096 41.75970225 34.81524215
 47.71040417 40.55436086 42.68101892 47.47907249 38.8961772  55.70701719
 39.70954528 20.65908188 53.35547931 34.39224602 53.052104   48.98191607
 35.55030533 22.10816837 43.13307997 34.38967561 52.91979567 39.66272299
 43.11733483 45.23886215 40.94738743 54.54875199 41.51965272 44.60026009
 39.75874274 21.99630292 33.65085485 38.4414015  21.08587786 44.49172478
 53.40463288 41.52916443 22.6183125  42.01100037 24.34711155 43.87002193
 46.7463016  26.19968145 43.75606925 27.05424841 26.01722749 37.62046021
 42.48095154 51.41982096 46.50330111 25.60228599 46.19782076 37.17512649
 47.9137163  42.81335122 22.53008236 33.40279523 47.40264507 42.02493538
 27.87336297 40.4401105  54.25888877 43.89549945 27.61935632 22.46859183
 28.89143909 16.03741312 39.98754177 51.7012972  30.16151618 39.74895738
```

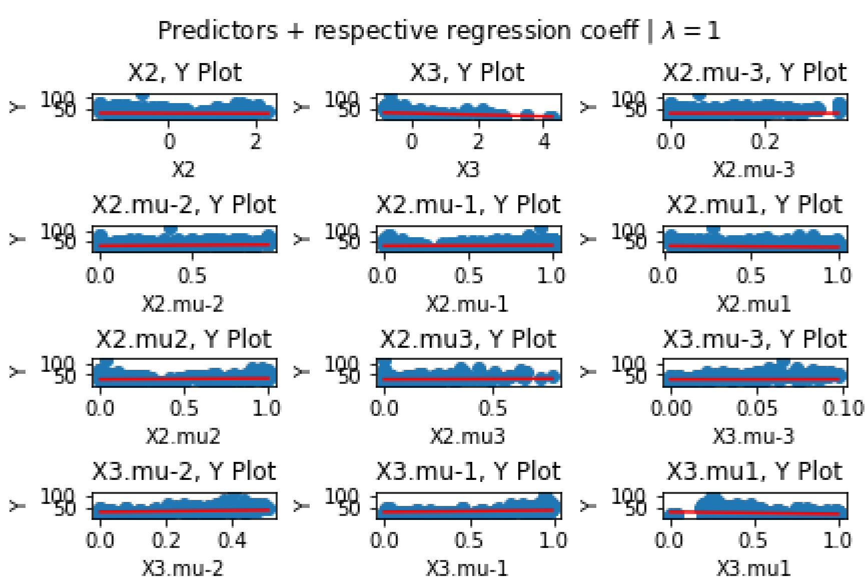
```
 30.95339666 36.49427139 41.80244351 51.03409909 42.40317287 38.78990729
 53.01882824 37.88242891 21.30803211 53.70273106 27.37870081 12.07357091
 50.98861046 51.7012972  42.55752207 30.00773369 21.22847851 30.07330239
 31.28648591 49.43371479 48.63951141 51.36364952 53.29408449 28.76760725
 44.88357174 45.81033629 33.25629489 44.31794877 37.91151607 22.23881086
 26.12532983 25.46522164 41.84267424 21.04923129 44.04281731 50.39510366
 44.24797291 52.35741922 48.61180188 28.66941084 26.42764159 54.44404268
 39.2659692  52.06745862 45.11666409 51.17559666 22.5678115  37.3816278
 21.48230865 45.84757408 53.70273106 23.34699128 26.3987651  48.01075786
 40.44020106 29.99829087 43.79500021 33.36629476 19.01171665 38.04226523
 34.30881883 33.02335551 26.42764159 47.78807307 37.50190582 28.25010845
 47.52634231 43.07285425 43.75606925 43.30900444 50.32276455 28.95987625
 21.85781404 22.86017307 53.06781049 40.9292812  51.53315056 52.61446954]
Uncertainties 0      13.252112
1      13.070722
2      13.138651
3      13.138651
4      13.264431
         ...
409    13.182845
410    13.250625
411    13.065114
412    13.263146
413    13.262030
Length: 414, dtype: float64
Intercept 29.002051715254527
Predicted vals [43.35897127 42.11912695 42.02203931 42.02203931 48.42170463 2
8.33389039
 37.6547829  42.02988233 13.37813082 25.69208313 40.41858193 51.24706104
 43.03779173 20.25525336 34.61655563 38.58103012 51.63461551 42.37540861
 42.54480831 54.11519205 29.02631105 46.92492782 31.55695123 47.13890335
 41.09764336 26.19971706 49.3606379  47.01021663 39.21425446 46.65922002
 16.69188961 34.7802038  40.43399214 43.2654077  45.18553329 21.74448774
 25.98591895 32.98464484 46.9963236  43.81554828 21.90018198 20.28834933
 39.44824843 38.99073393 47.71447694 39.99014275 39.44004351 37.88349265
 16.73234425 16.70772595 38.84705205 23.80414105 27.17702804 42.87840257
 43.86492572 30.28409662 40.47265665 52.93182559 16.8141562  44.75796934
 27.98018919 49.82334951 23.07227324 47.75629311 34.50641128 45.43511979
 52.42626313 49.96580112 38.6600939  42.44819857 51.09769418 37.75318645
 39.54874874 21.79474465 47.67408137 32.8238887  38.18583816 21.61916071
 39.75243393 29.33758362 43.17427385 39.76434142 46.94625729 18.24606955
 43.2566837  51.09342745 36.94183607 20.24041015 34.12378109 18.00025025
 51.85620333 34.06712443 20.18185451 30.49776929 45.19045952 50.22623013
 51.19742721 36.5873313  43.71737325 51.19742721 34.88975542 46.99667848
 52.38899681 52.61303935 39.98329211 51.63461551 44.48341672 32.87744625
 37.25528512 29.9971307  50.17434845 42.42850275 27.33062378 43.29116043
 42.44323064 36.42570483  8.79601898 21.77679727 25.28712183 43.62000168
 42.87840257 42.7196437  39.44582297 52.874301   47.24564902 52.38899681
 36.74689693 49.05382283 45.7845775  43.86057541 37.68751237 29.9971307
 38.26288596 41.97056766 42.56501228 33.71747866 45.12472538 44.79972853
 39.04708034 43.09104364 43.81554828 34.40145293 37.94223858 42.7196437
 33.03827872 48.96556101 52.874301   48.03554903 20.6265701  43.08856927
 43.47491224 48.50233748 32.98464484 47.86482197 20.24041015 21.79474465
 28.8703261  37.41595209 45.01734905 37.71599258 53.00548594 38.48534609
 20.67936954 49.96580112 52.874301   33.50165917 51.63461551 40.07303336
 37.83517366 29.11820304 16.9633461  49.14268453 51.09769418 42.57501172
 48.24835067 38.53514966 20.92106807 42.52618938 34.91436412 43.15811203
```

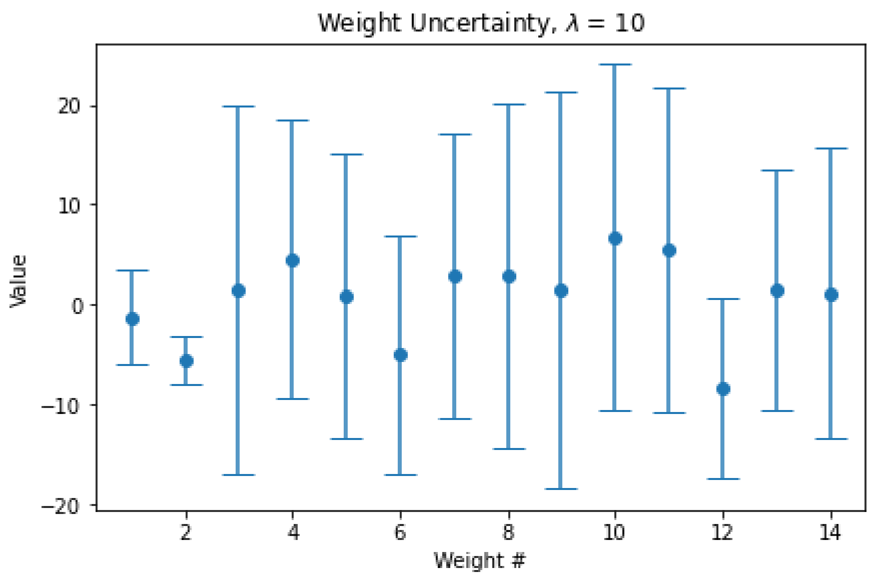
```
 16.773549   47.22575648 25.1162586  20.17913595 22.67228624 26.93648708
 21.47555929 24.66806304 42.93758179 20.53124173 37.98920344 39.73537292
 47.14676684 45.66327884 21.22814901 42.26720873 36.10608046 43.52714489
 43.06443588 40.93922492 34.55293706 42.12823511 38.66852331 44.11453222
 29.33758362 31.56562547 40.30514125 38.45762837 33.25296213 43.10080567
 48.32683762 51.85620333 25.63107094 51.29653579 25.60192221 41.32750752
 39.14131166 39.56303883 42.7196437  37.72890098 43.76503943 34.11991922
 39.09156757 34.06712443 41.28313151 52.38899681 20.41990497 41.41516566
 23.01702858 30.20115641 29.9971307  20.57334918 16.83905664 42.90059747
 27.62702262 45.97023355 49.26247333 40.05693479 40.1634934  36.19768622
 29.86505285 45.54607213 31.30711531 42.21305866 34.5428675  44.13824709
 42.52748489 32.10958957 33.67523277 11.39160594 41.64583125 30.29820643
 51.44397607 23.86163007 50.80706387 13.33775787 44.03833871 41.44188445
 51.63461551 36.37909201 29.03907908 22.92392377 43.96423788 30.04213919
 38.73354747 37.6152896  25.73757245 39.4487617  42.13424363 36.42510904
 47.07299593 41.16529279 43.03779173 46.73043779 39.30484803 54.11519205
 40.43584972 21.37216698 52.61303935 35.57649706 52.02388805 48.59601259
 36.86230592 22.48993822 43.30787727 35.47829631 51.44397607 40.39425569
 43.21639761 44.99083815 40.34437421 52.9756619  41.93271456 44.56875027
 40.04629125 22.5912046  35.23432758 38.47626543 20.32294885 43.2435453
 52.25841789 40.79419328 22.77696024 41.16352164 26.28392144 43.81554828
 46.10081923 23.92298653 43.71737325 28.80995356 27.70602327 38.66727687
 41.63951692 50.0702728  46.73405661 27.03786231 45.75759249 38.45550336
 47.5225065  41.9259892  21.82964043 35.17252551 46.69370545 41.63222354
 29.11402871 39.99014275 52.66386489 44.5891188  29.4412331  21.77679727
 30.39405499 16.724422   39.47547197 50.32966599 31.79048318 39.98007433
 31.53260129 37.1326704  41.30000571 50.57718262 41.65549896 39.84169725
 51.54142011 38.11133632 18.73790516 52.874301   28.68744564 10.74301561
 50.153283   50.32966599 42.93146739 29.79543936 20.40913231 29.95189626
 32.87744625 49.05382283 47.81399571 51.09342745 52.38899681 27.86573883
 43.54540226 44.03305968 35.04450104 43.1508256  38.01940493 22.53546708
 27.35432362 26.70775592 42.14934074 21.73101312 43.96423788 49.76090223
 43.03569518 51.85620333 48.23120508 30.86270384 27.85800662 52.82776817
 39.00893075 51.63461551 44.88595028 49.9188655  20.69050423 38.00218993
 20.62980484 45.16914511 52.874301   24.01161466 26.16100411 45.69640134
 40.18301696 30.41533486 42.44409092 35.13970308 17.8783897  39.01081496
 34.81609288 34.66968662 27.85800662 46.99667848 38.25192564 28.07860012
 46.80036989 42.28910326 43.71737325 42.96075885 49.9975442  28.05774608
 22.48214883 21.84742051 51.58987847 41.40894922 50.17434845 51.14763727]
Uncertainties 0      9.476926
1      9.403801
2      9.406595
3      9.406595
4      9.448968
         ...
409    9.489140
410    9.459311
411    9.397439
412    9.458192
413    9.460476
Length: 414, dtype: float64
Intercept 35.44506843703358
Predicted vals [37.98019333 37.98019338 37.98019338 37.98019338 37.98019349 3
7.98019307
 37.9801932  37.98019338 37.98019232 37.98019305 37.98019325 37.98019355
 37.9801934  37.98019288 37.98019324 37.98019321 37.98019355 37.98019339
 37.98019339 37.9801936  37.98019307 37.98019347 37.98019318 37.98019347
```

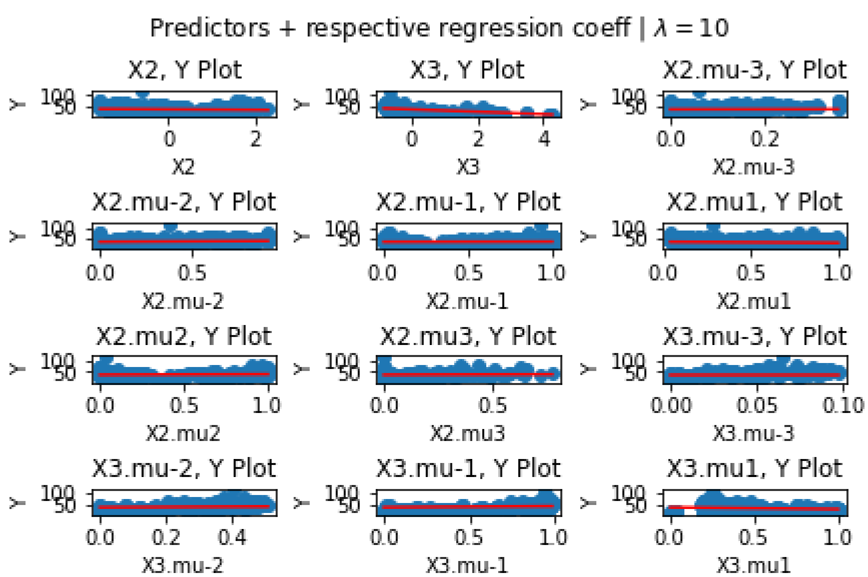
37.98019321	37.98019303	37.98019351	37.98019347	37.98019333	37.98019346
37.98019249	37.9801932	37.98019322	37.9801934	37.98019344	37.98019266
37.98019305	37.98019321	37.98019347	37.98019341	37.98019266	37.98019263
37.98019322	37.98019323	37.98019348	37.98019322	37.98019333	37.98019319
37.98019249	37.98019246	37.98019332	37.98019295	37.98019302	37.9801934
37.98019342	37.98019309	37.98019326	37.98019358	37.98019246	37.98019343
37.98019308	37.98019352	37.98019297	37.98019348	37.98019324	37.98019329
37.98019357	37.98019353	37.98019326	37.98019339	37.98019354	37.98019319
37.98019326	37.98019266	37.98019348	37.9801932	37.9801932	37.98019294
37.9801932	37.98019314	37.9801934	37.98019328	37.98019347	37.98019279
37.9801934	37.98019354	37.98019327	37.98019263	37.98019322	37.98019259
37.98019356	37.98019322	37.98019288	37.98019309	37.98019328	37.98019353
37.98019355	37.98019324	37.98019341	37.98019355	37.98019325	37.98019347
37.98019357	37.98019357	37.98019326	37.98019355	37.98019343	37.98019321
37.98019323	37.9801931	37.98019353	37.9801933	37.98019306	37.98019341
37.98019333	37.98019327	37.98019221	37.98019264	37.98019304	37.98019341
37.9801934	37.98019339	37.98019326	37.98019358	37.98019348	37.98019357
37.98019314	37.9801935	37.98019328	37.98019327	37.98019325	37.9801931
37.98019328	37.98019338	37.9801933	37.98019323	37.98019344	37.98019343
37.98019332	37.9801934	37.98019341	37.98019322	37.9801933	37.98019339
37.98019321	37.9801935	37.98019358	37.98019349	37.98019268	37.9801933
37.9801933	37.9801935	37.98019321	37.98019348	37.98019263	37.98019266
37.98019307	37.9801933	37.98019344	37.9801933	37.98019358	37.98019331
37.98019264	37.98019353	37.98019358	37.98019322	37.98019355	37.9801933
37.98019327	37.9801931	37.9801925	37.98019351	37.98019354	37.98019322
37.98019349	37.98019326	37.98019259	37.98019331	37.98019325	37.9801934
37.98019249	37.98019348	37.98019301	37.98019263	37.98019291	37.98019303
37.98019294	37.98019287	37.9801933	37.98019264	37.9801932	37.98019334
37.98019329	37.98019345	37.98019269	37.98019339	37.98019326	37.98019331
37.98019331	37.98019336	37.98019324	37.98019338	37.98019318	37.98019342
37.98019314	37.98019318	37.98019334	37.98019317	37.98019321	37.9801933
37.98019349	37.98019356	37.98019305	37.98019355	37.98019305	37.98019337
37.98019319	37.9801933	37.98019339	37.98019325	37.98019329	37.98019322
37.98019327	37.98019322	37.98019327	37.98019357	37.98019263	37.98019329
37.98019282	37.9801931	37.9801931	37.98019264	37.98019249	37.98019324
37.98019305	37.98019345	37.98019351	37.98019335	37.98019335	37.98019327
37.98019313	37.98019345	37.98019313	37.9801933	37.98019322	37.98019342
37.98019339	37.98019319	37.98019322	37.98019232	37.98019323	37.98019309
37.98019355	37.98019296	37.98019354	37.98019232	37.98019342	37.98019337
37.98019355	37.98019328	37.98019313	37.98019296	37.98019342	37.9801931
37.98019324	37.9801933	37.98019305	37.98019323	37.98019338	37.98019328
37.98019347	37.98019337	37.9801934	37.98019347	37.9801933	37.9801936
37.98019335	37.98019293	37.98019357	37.98019324	37.98019356	37.9801935
37.98019327	37.98019289	37.98019341	37.9801932	37.98019355	37.98019335
37.9801934	37.98019344	37.98019322	37.98019358	37.98019338	37.98019343
37.98019332	37.98019296	37.98019325	37.98019321	37.98019263	37.98019332
37.98019356	37.98019322	37.98019295	37.98019323	37.98019307	37.98019341
37.98019346	37.98019286	37.98019341	37.98019307	37.9801931	37.98019331
37.98019327	37.98019353	37.98019346	37.98019308	37.98019345	37.98019332
37.98019348	37.98019328	37.98019264	37.98019325	37.98019347	37.98019333
37.98019312	37.98019322	37.98019357	37.98019342	37.98019314	37.98019264
37.98019309	37.98019249	37.98019318	37.98019353	37.98019314	37.98019331
37.98019315	37.98019322	37.9801933	37.98019353	37.98019328	37.98019334
37.98019355	37.98019322	37.98019266	37.98019358	37.98019311	37.9801923
37.98019353	37.98019353	37.9801934	37.98019309	37.98019286	37.9801931
37.98019321	37.9801935	37.98019349	37.98019354	37.98019357	37.98019303
37.98019333	37.98019325	37.98019325	37.98019333	37.9801932	37.98019295

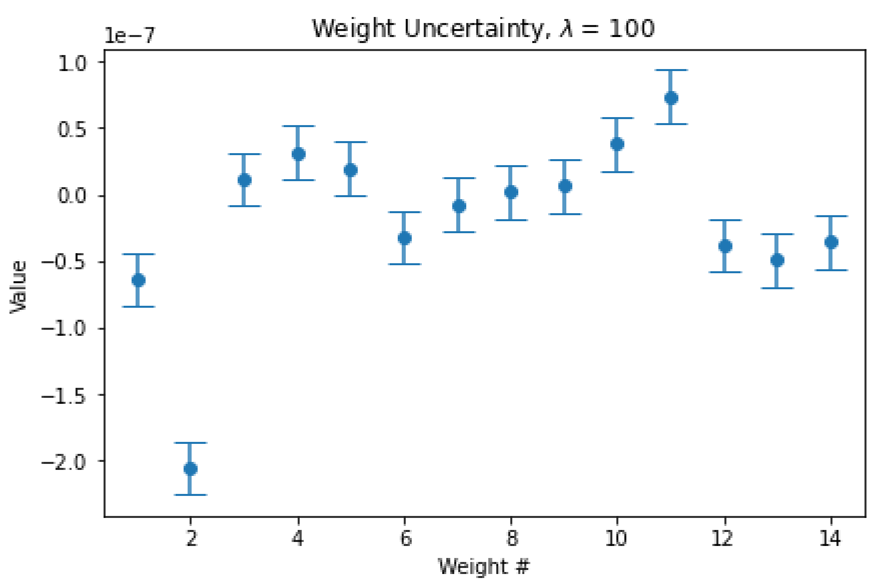
```
 37.98019308 37.98019307 37.98019338 37.98019294 37.98019342 37.98019352
 37.98019331 37.98019356 37.98019349 37.98019317 37.9801931  37.98019357
 37.9801932  37.98019355 37.98019343 37.98019352 37.98019272 37.98019326
 37.98019264 37.98019344 37.98019358 37.980193   37.98019302 37.98019328
 37.98019327 37.98019312 37.98019321 37.98019325 37.9801925  37.98019332
 37.98019312 37.98019324 37.9801931  37.98019347 37.98019328 37.98019306
 37.98019347 37.98019332 37.98019341 37.98019339 37.98019352 37.98019303
 37.98019296 37.98019266 37.98019355 37.98019337 37.98019353 37.98019355]
Uncertainties 0      13.590045
1       13.590045
2       13.590045
3       13.590045
4       13.590045
          ...
409     13.590045
410     13.590045
411     13.590045
412     13.590045
413     13.590045
Length: 414, dtype: float64
Intercept 37.980193203288984
```

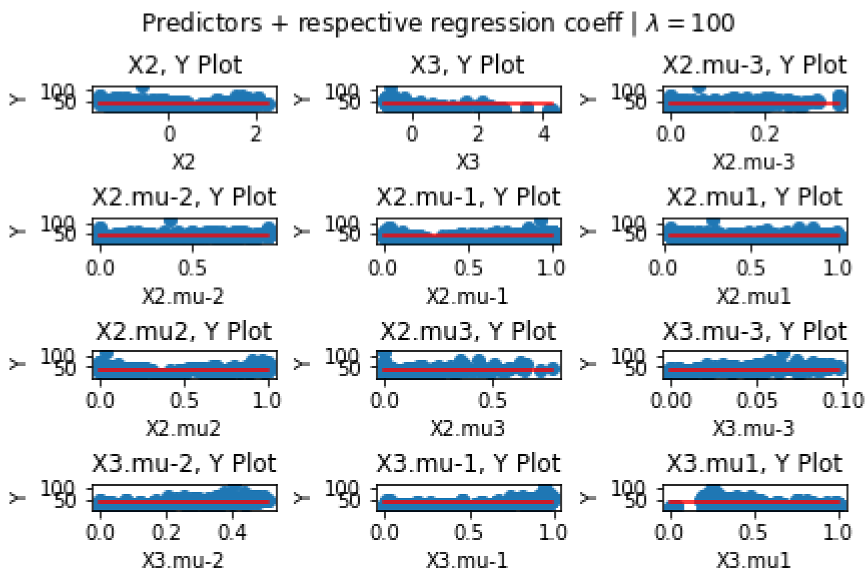





Weight Uncertainty, λ = 10
Value
Weight #
20
10
0
−10
−20
2
4
6
8
10
12
14


Predictors + respective regression coeff | λ = 10
X2, Y Plot
X3, Y Plot
X2.mu-3, Y Plot
X2.mu-2, Y Plot
X2.mu-1, Y Plot
X2.mu1, Y Plot
X2.mu2, Y Plot
X2.mu3, Y Plot
X3.mu-3, Y Plot
X3.mu-2, Y Plot
X3.mu-1, Y Plot
X3.mu1, Y Plot
X2
X3
X2.mu-3
X2.mu-2
X2.mu-1
X2.mu1
X2.mu2
X2.mu3
X3.mu-3
X3.mu-2
X3.mu-1
X3.mu1
Y


Weight Uncertainty, λ = 100
1e−7
Value
Weight #
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2
4
6
8
10
12
14



In [ ]:
```
# 5. Compare linear regression model, gaussian basis model
BayesianRidge().fit(X=predictors, y=dfr["Y"])
```

In [ ]:
```
# 6. Cross Validation
from sklearn.metrics import make_scorer

rmse = make_scorer(mean_squared_error)
print(cross_val_score(bay_1, X=predictors, y=dfr["Y"], cv=10, scoring=rmse))
print(cross_val_score(bay_2, X=predictors, y=dfr["Y"], cv=10, scoring=rmse))
print(cross_val_score(bay_3, X=predictors, y=dfr["Y"], cv=10, scoring=rmse))

print(cross_val_score(stan_1, X=standardized_pred, y=dfr["Y"], cv=10, scoring=
rmse))
print(cross_val_score(stan_2, X=standardized_pred, y=dfr["Y"], cv=10, scoring=
rmse))
print(cross_val_score(stan_3, X=standardized_pred, y=dfr["Y"], cv=10, scoring=
rmse))
```

```
[ 63.89114084  56.89993407  98.38583199  94.33255073  36.11357478
  83.65535406 213.3551035   82.54990061  59.33830889  64.5932842 ]
[ 77.88860063  87.40388346 124.81751986  99.90442721  54.60008824
 102.32613294 222.84041319 107.13168165  59.16076487  81.85398287]
[159.2164679  180.81317535 212.84521591 168.61501452 144.33281553
 176.98253728 281.3658193  186.24846372 116.48008443 169.07217252]
[ 68.87284554  69.19693126 110.71281546  96.49990031  46.60044329
 104.61997928 210.19846791 101.07330672  65.72946081  74.38563569]
[ 68.54828677  71.12478252 110.89243952  97.23924919  47.08919486
 103.89155106 210.71163211 101.91304775  64.47974023  75.90370003]
[165.46161974 187.04141041 219.67453442 173.68079031 150.93929919
 183.30308791 285.77624639 192.15005657 121.26635377 174.9351637 ]
```

Classification Problem (25 pts)

In [ ]:
```
# Dataset Init
dfc = pd.read_excel("ENB2012_data.xlsx")
dfc["Y1-Y2"] = np.where(dfc["Y1"] - dfc["Y2"] > 0, 1, 0)
dfc.head()
```

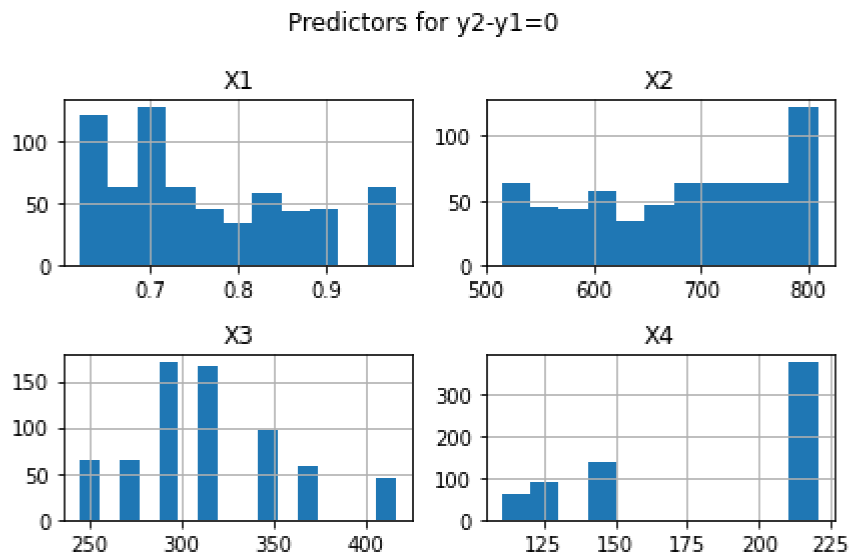
Out[ ]:

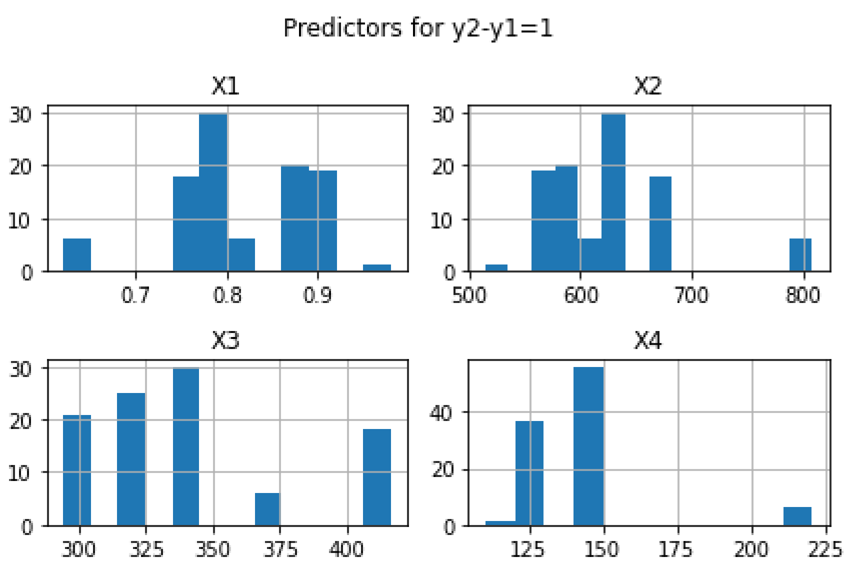
	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2	Y1-Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33	0
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33	0
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33	0
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33	0
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28	0

In [ ]:
```
# 1. Visualization

excl_hist = ["Y1", "Y2", "Y1-Y2", "X5", "X6", "X7", "X8"]

# Histogram of predictors X1, X2, X3, X4 that have label 0
fig_hist_0, ax_hist_0 = plt.subplots()
dfc[dfc["Y1-Y2"] == 0].drop(excl_hist, axis=1).hist(bins="auto", ax=ax_hist_0)
fig_hist_0.suptitle("Predictors for y2-y1=0")
fig_hist_0.tight_layout()

# Histogram of predictors X1, X2, X3, X4 that have label 1
fig_hist_1, ax_hist_1 = plt.subplots()
dfc[dfc["Y1-Y2"] == 1].drop(excl_hist, axis=1).hist(bins="auto", ax=ax_hist_1)
fig_hist_1.suptitle("Predictors for y2-y1=1")
fig_hist_1.tight_layout()

# Construct conditional PMFs
x_discrete = ["X5", "X6", "X7", "X8"]

# P(Xn | Y=0)
fig_pmf_0, ax_pmf_0 = plt.subplots(2, 2)
for i in range(len(x_discrete)):
    data = dfc[dfc["Y1-Y2"] == 0][x_discrete[i]]
    data = data.value_counts().sort_index()
    data = data.divide(sum(data.values))
    # print(data)
    x = data.index
    p = data.values
    ax_pmf_0[lin_to_mat(2, i)].plot(x, p, "bo", ms=8, mec="b")
    ax_pmf_0[lin_to_mat(2, i)].vlines(x, 0, p, linestyles="-", lw=2)
    ax_pmf_0[lin_to_mat(2, i)].set_title(x_discrete[i])
    ax_pmf_0[lin_to_mat(2, i)].set_ylim([0, 0.6])
    ax_pmf_0[lin_to_mat(2, i)].set_xticks(x)
fig_pmf_0.suptitle("Conditional PMFs for Y=0")

fig_pmf_0.tight_layout()


# P(Xn | Y=1)
fig_pmf_1, ax_pmf_1 = plt.subplots(2, 2)
for i in range(len(x_discrete)):
    data = dfc[dfc["Y1-Y2"] == 1][x_discrete[i]]
    data = data.value_counts().sort_index()
    data = data.divide(sum(data.values))
    # print(data)
    x = data.index
    p = data.values
    ax_pmf_1[lin_to_mat(2, i)].plot(x, p, "bo", ms=8, mec="b")
    ax_pmf_1[lin_to_mat(2, i)].vlines(x, 0, p, linestyles="-", lw=2)
    ax_pmf_1[lin_to_mat(2, i)].set_title(x_discrete[i])
    ax_pmf_1[lin_to_mat(2, i)].set_ylim([0, max(0.6, max(p))])
    ax_pmf_1[lin_to_mat(2, i)].set_xticks(x)
fig_pmf_1.suptitle("Conditional PMFs for Y=1")
fig_pmf_1.tight_layout()

_="""
```

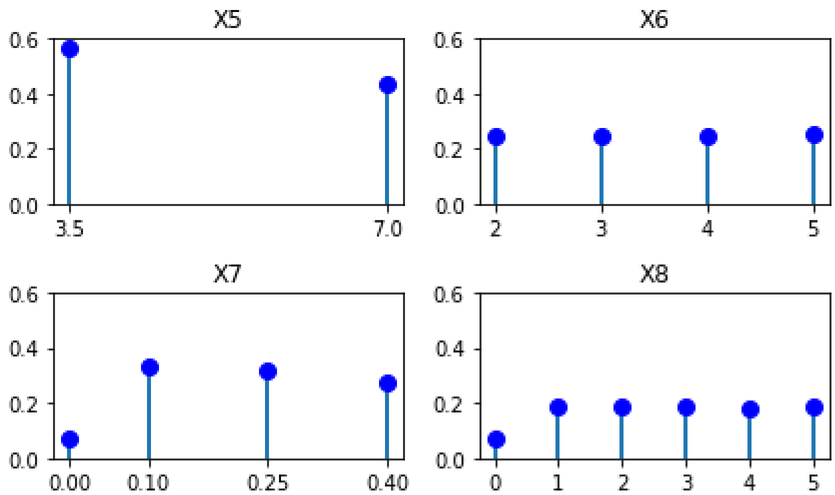
```
X5 and X7 appear to be very good predictors of the class labels, looking at th
eir conditional distributions.
"""
```

```
C:\Users\SAADMU~1\AppData\Local\Temp/ipykernel_28320/1939464138.py:7: UserWar
ning: To output multiple subplots, the figure containing the passed axes is b
eing cleared
  dfc[dfc["Y1-Y2"] == 0].drop(excl_hist, axis=1).hist(bins="auto", ax=ax_hist
_0)
C:\Users\SAADMU~1\AppData\Local\Temp/ipykernel_28320/1939464138.py:13: UserWa
rning: To output multiple subplots, the figure containing the passed axes is
being cleared
  dfc[dfc["Y1-Y2"] == 1].drop(excl_hist, axis=1).hist(bins="auto", ax=ax_hist
_1)
```

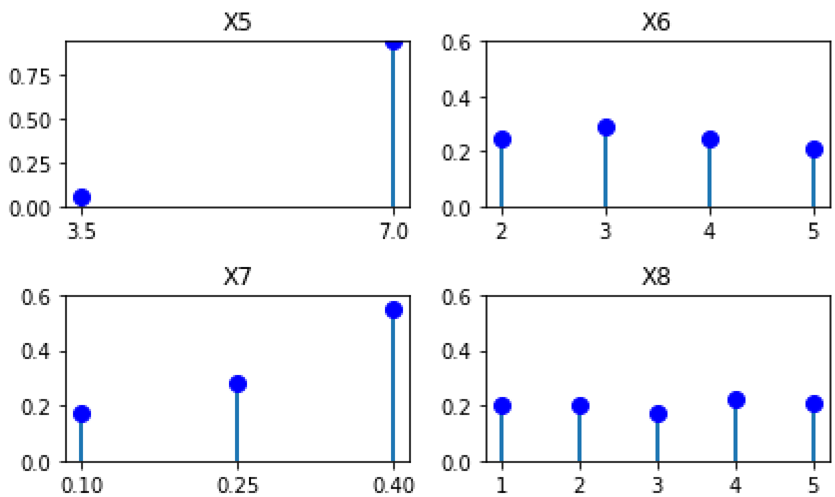




Conditional PMFs for Y=0



Conditional PMFs for Y=1



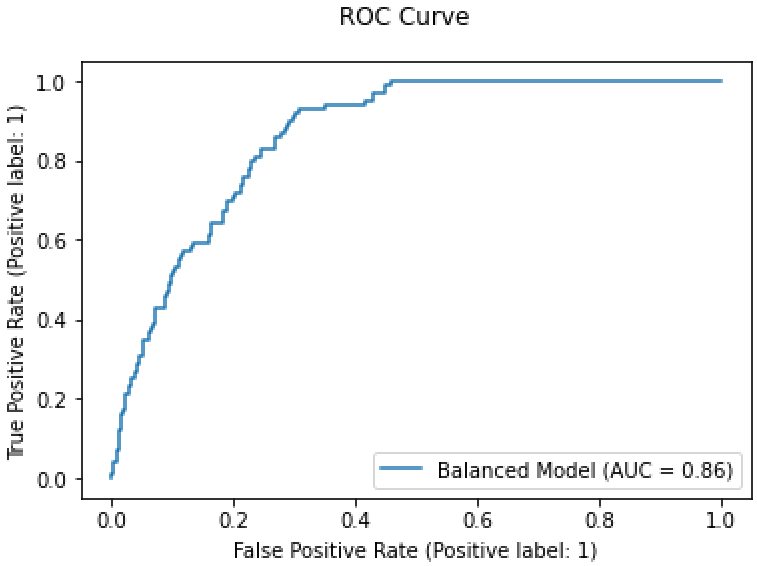
In [ ]:
```python
# 2. Logistic Regression
pred_c_names = ["X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8"]
pred_c_data = dfc[pred_c_names]
logistic_model = LogisticRegression(max_iter=500).fit(X=pred_c_data, y=dfc["Y1
-Y2"])

prob_preds = logistic_model.predict_proba(X=pred_c_data)[:,1]
show_weights(pred_c_names, logistic_model.coef_[0])

# Assuming Y=1 is "positive", Y=0 is "negative"
preds = logistic_model.predict(pred_c_data)
preds_df = pd.DataFrame(preds.transpose(), columns=["preds"])
true_pos = len(dfc[(dfc["Y1-Y2"] == 1) & (preds_df["preds"] == 1)])

print("\nAccuracy\nRecall: ", true_pos / len([i for i in preds if i == 1]))
print("Precision: ", true_pos / len(dfc[dfc[pred_c_names] == 1]))
print("Model score:", logistic_model.score(X=pred_c_data, y=dfc["Y1-Y2"]))
```

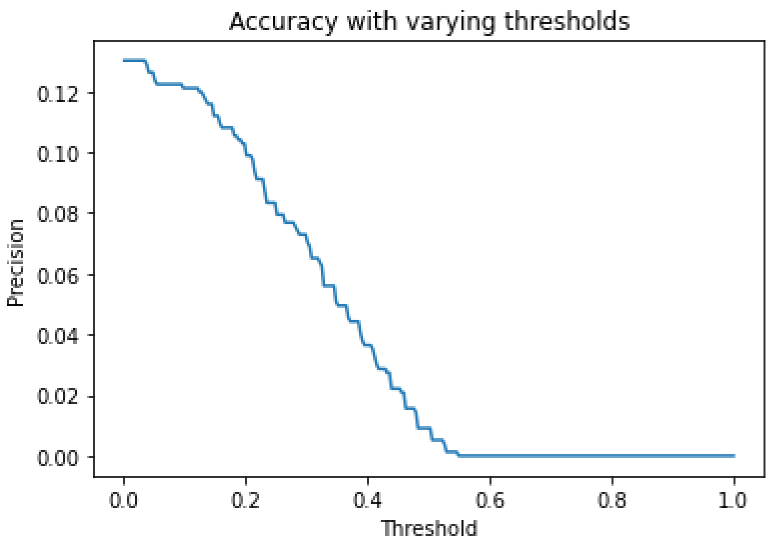
```
X1 weight: 0.1314098373909968
X2 weight: 0.01376272663718662
X3 weight: -0.0019824349199593157
X4 weight: 0.007872580757269867
X5 weight: 1.6519035109073805
X6 weight: -0.08958701981032652
X7 weight: 3.1448815458541177
X8 weight: 0.08805294156710261

Accuracy
Recall:  0.5833333333333334
Precision:  0.009114583333333334
Model score: 0.8723958333333334
```

In [ ]:

```
# 3. ROC Curve
fig_roc, ax_roc = plt.subplots()
fig_roc.suptitle("ROC Curve")
RocCurveDisplay.from_predictions(y_true=dfc["Y1-Y2"], y_pred=prob_preds, ax=ax
_roc, name="Balanced Model")

fig_thresh, ax_thresh = plt.subplots()

threshs = np.linspace(0.001, 1, 300)
prec_data = []
for thresh in threshs:
    preds_df["thresh"] = [1 if i > thresh else 0 for i in prob_preds]
    true_pos = len(dfc[(dfc["Y1-Y2"] == 1) & (preds_df["thresh"] == 1)])
    prec_data.append(true_pos / len(dfc[dfc[pred_c_names] == 1]))


len(dfc[(dfc["Y1-Y2"] == 1) & (preds_df["preds"] == 1)])
ax_thresh.plot(threshs, prec_data)
ax_thresh.set_title("Accuracy with varying thresholds")
ax_thresh.set_ylabel("Precision")
ax_thresh.set_xlabel("Threshold")
```

```
Out[ ]: Text(0.5, 0, 'Threshold')
```





In [ ]:
```python
# 4. Bayesian Regression
def bay_logistic(alpha):
    model = BernoulliNB(alpha=alpha, fit_prior=True)
    model.fit(X=pred_c_data, y=dfc["Y1-Y2"])
    model.predict_proba(pred_c_data)
    weights = model.coef_
    print("Weights", weights)
    # print("Weight uncertainties", )
    return model, weights


a_1, w1 = bay_logistic(0.1)
a_2, w2 = bay_logistic(1)
a_3, w3 = bay_logistic(10)
a_4, w4 = bay_logistic(100)
```

```
Weights [[-0.0009985 -0.0009985 -0.0009985 -0.0009985 -0.0009985 -0.0009985
  -0.0009985 -0.0009985]]
Weights [[-0.0098523 -0.0098523 -0.0098523 -0.0098523 -0.0098523 -0.0098523
  -0.0098523 -0.0098523]]
Weights [[-0.08701138 -0.08701138 -0.08701138 -0.08701138 -0.08701138 -0.0870
1138
  -0.08701138 -0.08701138]]
Weights [[-0.40546511 -0.40546511 -0.40546511 -0.40546511 -0.40546511 -0.4054
6511
  -0.40546511 -0.40546511]]

C:\Python39\lib\site-packages\sklearn\utils\deprecation.py:103: FutureWarnin
g: Attribute `coef_` was deprecated in version 0.24 and will be removed in 1.
1 (renaming of 0.26).
  warnings.warn(msg, category=FutureWarning)
C:\Python39\lib\site-packages\sklearn\utils\deprecation.py:103: FutureWarnin
g: Attribute `coef_` was deprecated in version 0.24 and will be removed in 1.
1 (renaming of 0.26).
  warnings.warn(msg, category=FutureWarning)
C:\Python39\lib\site-packages\sklearn\utils\deprecation.py:103: FutureWarnin
g: Attribute `coef_` was deprecated in version 0.24 and will be removed in 1.
1 (renaming of 0.26).
  warnings.warn(msg, category=FutureWarning)
C:\Python39\lib\site-packages\sklearn\utils\deprecation.py:103: FutureWarnin
g: Attribute `coef_` was deprecated in version 0.24 and will be removed in 1.
1 (renaming of 0.26).
  warnings.warn(msg, category=FutureWarning)
```

In [ ]:
```python
# 5. Cross validation
print(cross_val_score(estimator=a_1, X=pred_c_data, y=dfc["Y1-Y2"], cv=10))
print(cross_val_score(estimator=a_2, X=pred_c_data, y=dfc["Y1-Y2"], cv=10))
print(cross_val_score(estimator=a_3, X=pred_c_data, y=dfc["Y1-Y2"], cv=10))
print(cross_val_score(estimator=a_4, X=pred_c_data, y=dfc["Y1-Y2"], cv=10))

_="""
The model accuracy does not appear to change with varying values of alpha
"""
```

```
[0.24675325 0.87012987 0.87012987 0.87012987 0.87012987 0.87012987
 0.87012987 0.87012987 0.86842105 0.86842105]
[0.24675325 0.87012987 0.87012987 0.87012987 0.87012987 0.87012987
 0.87012987 0.87012987 0.86842105 0.86842105]
[0.24675325 0.87012987 0.87012987 0.87012987 0.87012987 0.87012987
 0.87012987 0.87012987 0.86842105 0.86842105]
[0.87012987 0.87012987 0.87012987 0.87012987 0.87012987 0.87012987
 0.87012987 0.87012987 0.86842105 0.86842105]
```

In [ ]:
```python
# 6.

x1=np.array([0.8,600.0,286.0,138.1,5,4,0.25,0]).reshape((1, 8))
x2=np.array([0.67,630.0,296.0,238.1,2,6,0.5,3]).reshape((1, 8))

for w in [w1, w2, w3, w4]:
    print((w * x1 + w * x2) / 2)
    # print(m.predict_proba(x2))
```

```
[[-7.33899212e-04 -6.14078933e-01 -2.90564178e-01 -1.87818288e-01
  -3.49475815e-03 -4.99251165e-03 -3.74438374e-04 -1.49775349e-03]]
[[-7.24143789e-03 -6.05916231e+00 -2.86701826e+00 -1.85321696e+00
  -3.44830376e-02 -4.92614822e-02 -3.69461117e-03 -1.47784447e-02]]
[[-6.39533621e-02 -5.35119968e+01 -2.53203107e+01 -1.63668400e+01
  -3.04539819e-01 -4.35056885e-01 -3.26292664e-02 -1.30517065e-01]]
[[-2.98016854e-01 -2.49361041e+02 -1.17990346e+02 -7.62679868e+01
  -1.41912788e+00 -2.02732554e+00 -1.52049416e-01 -6.08197662e-01]]
```

In [ ]: