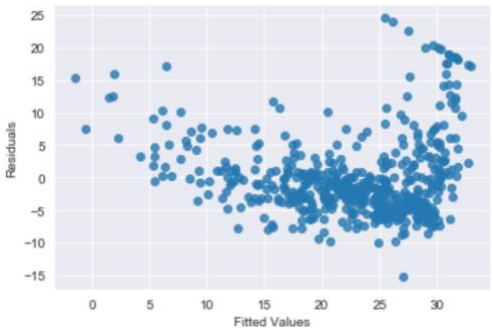
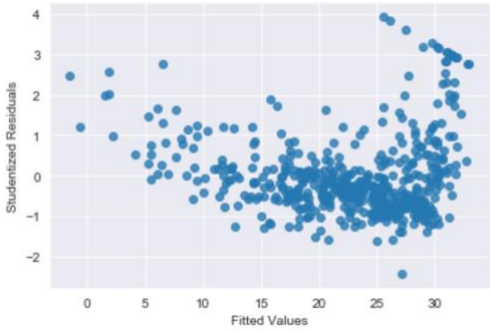
Homework 1

Question 1

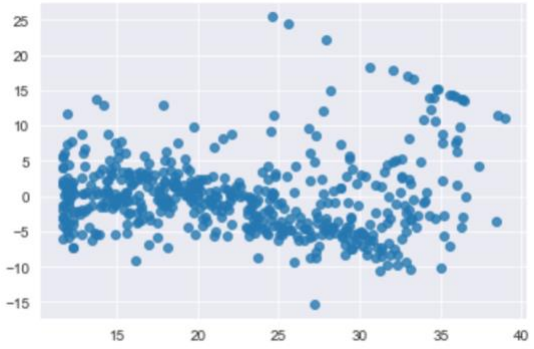
Plots in this question are all from the Jupyter Notebook, "Homework 1 Question 1 notebook help", provided in Courseworks.

1. With only $'lstat'$ as the predictor, both 'Residuals vs. Fitted values' and 'Studentized Residuals vs. Fitted values' show a clear pattern below. It looks like a non-linear relationship.





After adding the squared term '$lstat^2$', the picture below shows no discernible pattern, with much more evenly distributed 'residuals' (y-axis) on each value of 'fitted values' (x-axis). Hence, this is a much better model that meets the assumptions for linear regression.



2. No, it does not correspond to the one in ISLR. Because the function 'poly(lstat, 5)' works differently in R than 'PolynomialFeatures(degree=5)' in Python-sklearn, when transforming into a fifth-order polynomial.
   In Python, it transforms 'lstat' into a vector, $[1,\ lstat,\ lstat^2,\ lstat^3,\ lstat^4,\ lstat^5]$.

```
boston.lstat.values.reshape(-1,1)
```

```
array([[ 4.98],
       [ 9.14],
       [ 4.03],
       [ 2.94],
       [ 5.33],
       [ 5.21],
       [12.43],
       [19.15],
       [29.93],
       [17.1 ],
       [20.45],
       [13.27],
```

```
from sklearn.preprocessing import PolynomialFeatures
polynomial_features= PolynomialFeatures(degree=5)
x = boston.lstat.values.reshape(-1,1)
xp = polynomial_features.fit_transform(x)
```

```
xp
```

```
array([[1.00000000e+00, 4.98000000e+00, 2.48004000e+01, 1.23505992e+02,
        6.15059840e+02, 3.06299800e+03],
       [1.00000000e+00, 9.14000000e+00, 8.35396000e+01, 7.63551944e+02,
        6.97886477e+03, 6.37868240e+04],
       [1.00000000e+00, 4.03000000e+00, 1.62409000e+01, 6.54508270e+01,
        2.63766833e+02, 1.06298034e+03],
       ...,
       [1.00000000e+00, 5.64000000e+00, 3.18096000e+01, 1.79406144e+02,
        1.01185065e+03, 5.70683768e+03],
       [1.00000000e+00, 6.48000000e+00, 4.19904000e+01, 2.72097792e+02,
        1.76319369e+03, 1.14254951e+04],
       [1.00000000e+00, 7.88000000e+00, 6.20944000e+01, 4.89303872e+02,
        3.85571451e+03, 3.03830303e+04]])
```

In R, the 'poly()' function creates a polynomial within the 'lm()' model function, and yields different values for the features vector.

```
> poly(lstat,5)
                1           2           3           4           5
 [1,] -4.781458e-02  3.895455e-02 -1.906462e-02 -5.439764e-03  2.971418e-02
 [2,] -2.189160e-02 -1.078168e-02  3.564769e-02 -3.320731e-02  3.243484e-03
 [3,] -5.373449e-02  5.360400e-02 -4.390947e-02  2.594825e-02 -1.748063e-05
 [4,] -6.052681e-02  7.191994e-02 -7.909404e-02  7.895767e-02 -6.798137e-02
 [5,] -4.563356e-02  3.386588e-02 -1.119724e-02 -1.399341e-02  3.535552e-02
 [6,] -4.638134e-02  3.559186e-02 -1.381903e-02 -1.122991e-02  3.369972e-02
 [7,] -1.390015e-03 -3.349791e-02  3.181157e-02  3.770708e-03 -3.740751e-02
 [8,]  4.048556e-02 -3.428901e-02 -3.469467e-02  3.878873e-02  3.176056e-02
 [9,]  1.076610e-01  9.234906e-02 -1.539020e-02 -1.016926e-01 -9.071707e-02
[10,]  2.771102e-02 -4.053863e-02 -1.170131e-02  4.297099e-02 -3.205767e-03
[11,]  4.858649e-02 -2.737309e-02 -4.791903e-02  2.822853e-02  5.146088e-02
[12,]  3.844432e-03 -3.694564e-02  2.629519e-02  1.415648e-02 -3.935597e-02
[13,]  1.904925e-02 -4.153420e-02  3.700070e-03  3.729857e-02 -2.336605e-02
[14,] -2.737531e-02 -2.217570e-03  3.051285e-02 -3.783175e-02  1.819934e-02
[15,] -1.491234e-02 -2.016271e-02  3.806211e-02 -2.286674e-02 -1.507927e-02
[16,] -2.606669e-02 -4.356675e-03  3.201571e-02 -3.707992e-02  1.471611e-02
[17,] -3.784420e-02  1.704809e-02  1.162559e-02 -3.335963e-02  3.862129e-02
[18,]  1.256851e-02 -4.056561e-02  1.428469e-02  2.914584e-02 -3.381932e-02
```

Question 2

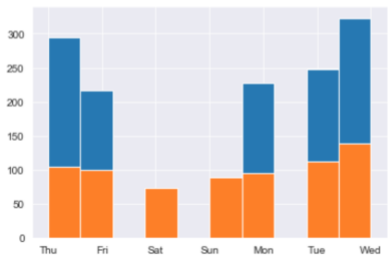
1. The dependent variable here is 'spam', since the goal is to predict a future email is spam or not.

2. Report relationships through visualization.

'day of week' group by 'spam':

```
# relationship between 'day of week' and 'spam'
spam['day of week'].groupby(spam['spam']).hist()
```

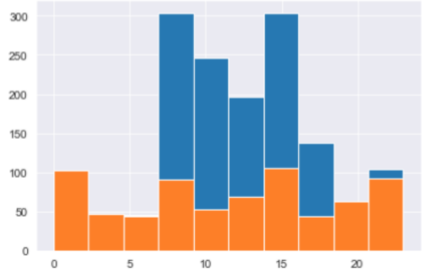
```
spam
no     AxesSubplot(0.125,0.125;0.775x0.755)
yes    AxesSubplot(0.125,0.125;0.775x0.755)
Name: day of week, dtype: object
```



'time of day' group by 'spam':

```
spam['time of day'].groupby(spam['spam']).hist()
```

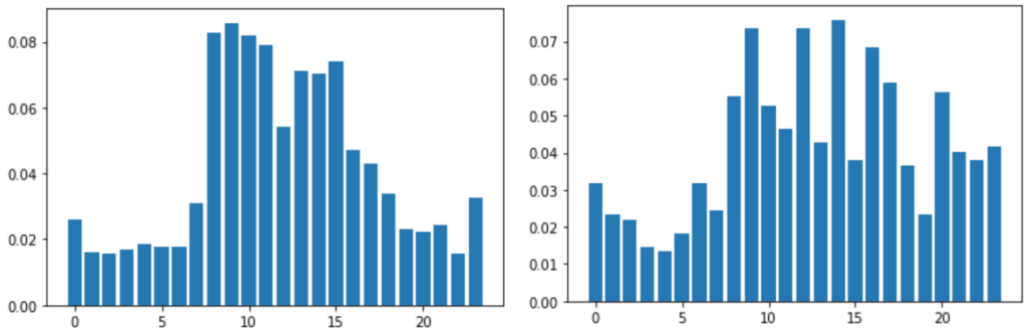
```
spam
no     AxesSubplot(0.125,0.125;0.775x0.755)
yes    AxesSubplot(0.125,0.125;0.775x0.755)
Name: time of day, dtype: object
```



3.

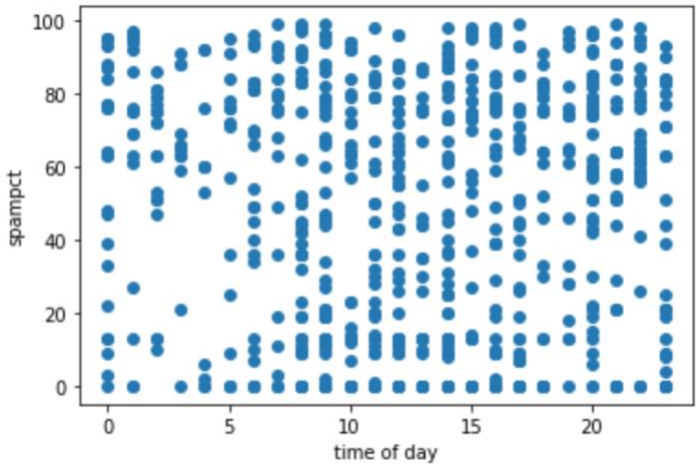
a. ‘spampct’ has 1353 missing values.

b. The plot on the left is the distribution of ‘time of day’ where ‘spampct’ is missing, and on the right is where ‘spampct’ is available. The y-axis corresponds to the percentage of total emails in each particular hour of the day.



When ‘spampct’ is missing, it follows more like a Normal Distribution, with the exception 12:00 pm being very low, but overall the emails during the day are much more frequent than during the night.
When ‘spampct’ is available, the distribution becomes more random as the percentage of total emails consistently goes up and down.

c. Below is the scatter plot. There are 1451 unique points plotted.



One way to solve the problem of overplotting is to use ‘Bubble Plot’, which makes the size of a unique point larger if it is plotted repeatedly.

Question 3

a) Code in zz2732_hw1.ipynb
mean = 0, variance = 1

```
3 np.random.seed(1)
4 X = np.random.normal(0, 1, 100)
5 X
```

```
array([ 1.62434536, -0.61175641, -0.52817175, -1.07296862,  0.86540763,
       -2.3015387 ,  1.74481176, -0.7612069 ,  0.3190391 , -0.24937038,
        1.46210794, -2.06014071, -0.3224172 , -0.38405435,  1.13376944,
       -1.09989127, -0.17242821, -0.87785842,  0.04221375,  0.58281521,
       -1.10061918,  1.14472371,  0.90159072,  0.50249434,  0.90085595,
       -0.68372786, -0.12289023, -0.93576943, -0.26788808,  0.53035547,
       -0.69166075, -0.39675353, -0.6871727 , -0.84520564, -0.67124613,
       -0.0126646 , -1.11731035,  0.2344157 ,  1.65980218,  0.74204416,
       -0.19183555, -0.88762896, -0.74715829,  1.6924546 ,  0.05080775,
       -0.63699565,  0.19091548,  2.10025514,  0.12015895,  0.61720311,
        0.30017032, -0.35224985, -1.1425182 , -0.34934272, -0.20889423,
        0.58662319,  0.83898341,  0.93110208,  0.28558733,  0.88514116,
       -0.75439794,  1.25286816,  0.51292982, -0.29809284,  0.48851815,
       -0.07557171,  1.13162939,  1.51981682,  2.18557541, -1.39649634,
       -1.44411381, -0.50446586,  0.16003707,  0.87616892,  0.31563495,
       -2.02220122, -0.30620401,  0.82797464,  0.23009474,  0.76201118,
       -0.22232814, -0.20075807,  0.18656139,  0.41005165,  0.19829972,
        0.11900865, -0.67066229,  0.37756379,  0.12182127,  1.12948391,
        1.19891788,  0.18515642, -0.37528495, -0.63873041,  0.42349435,
        0.07734007, -0.34385368,  0.04359686, -0.62000084,  0.69803203])
```

b) Code in zz2732_hw1.ipynb
mean = 0, variance = 0.25

```
3 eps = np.random.normal(0, 0.25**0.5, 100)
4 eps
```

```
array([-0.22356428,  0.61225385,  0.20174582,  0.29678926, -0.54745592,
        0.08469122,  0.37027823, -0.4768503 , -0.13310925,  0.01630727,
       -0.68655866,  0.1575797 ,  0.42308032, -0.42975797,  0.17527299,
       -0.65614171, -0.01934775, -0.80788618,  0.56070885,  0.20445027,
       -0.01230848, -0.38758081,  0.63687797,  0.98355087, -0.92899093,
        0.61808202,  0.81382538,  0.16900585, -0.59963402,  0.43167266,
       -0.09046015, -0.30196031, -0.61502907,  0.27526875,  0.39640343,
       -0.31176536,  0.26028817, -0.57217069,  0.40093052,  0.02328365,
       -0.09328489, -0.05087294,  0.43444308,  0.37520582,  0.26473266,
        0.0688506 ,  0.03891056,  0.30919013,  0.11624728,  0.3412757 ,
       -0.15505839, -1.21741888,  0.5194123 ,  1.09348982,  0.22068222,
       -0.05007762, -0.06822237, -0.05952709,  0.0087047 , -0.56100936,
       -0.25854723, -0.49851341,  0.12439958, -0.14832058,  0.24760566,
       -0.08735158,  0.49316759,  0.10676695,  1.09534986, -0.94818046,
       -0.32345834,  0.45074345,  1.26416285, -0.12431739,  0.0218345 ,
       -0.11315712,  0.66572856, -0.14365393,  0.34003492, -0.1599008 ,
       -0.63627938,  0.15677386,  0.25159241,  0.64661294, -0.05522351,
       -0.30868103,  0.28138055,  0.12036855,  0.14033254, -0.03655635,
        0.58016928,  0.18474636,  0.95232935,  0.55552835,  0.3295249 ,
       -0.81371917,  0.30115964,  0.2101411 ,  0.40547584,  0.52222105])
```

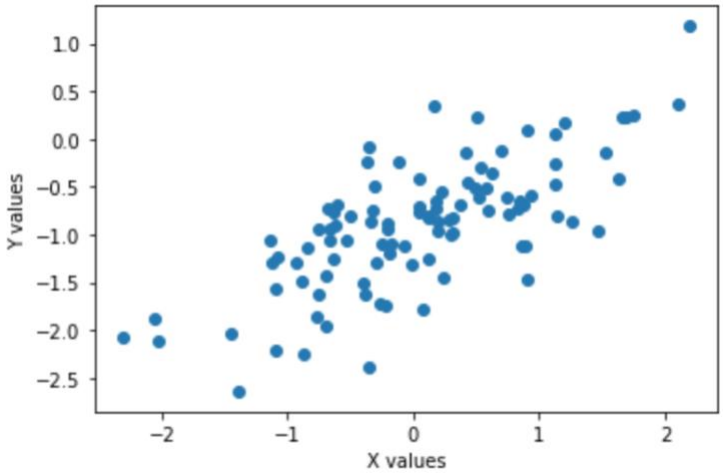
c) Model Y.

```
3 Y = -1 + 0.5*X + eps
4 Y
```

```
array([-0.4113916 , -0.69362435, -1.06234006, -1.23969505, -1.11475211,
       -2.06607813,  0.24268411, -1.85745375, -0.9735897 , -1.10837791,
       -0.95550469, -1.87249066, -0.73812828, -1.62178515, -0.25784229,
       -2.20608734, -1.10556186, -2.24681539, -0.41818427, -0.50414212,
       -1.56261807, -0.81521895,  0.08767333,  0.23479804, -1.47856296,
       -0.72378191, -0.24761974, -1.29887887, -1.73357806, -0.30314961,
       -1.43629053, -1.50033708, -1.95861542, -1.14733407, -0.93921963,
       -1.31809766, -1.29836701, -1.45496285,  0.2308316 , -0.60569427,
       -1.18920266, -1.49468742, -0.93913607,  0.22143312, -0.70986346,
       -1.24964722, -0.86563169,  0.3593177 , -0.82367324, -0.35012274,
       -1.00497323, -2.39354381, -1.0518468 , -0.08118154, -0.88376489,
       -0.75676602, -0.64873067, -0.59397605, -0.84850163, -1.11843878,
       -1.6357462 , -0.87207934, -0.61913551, -1.29736699, -0.50813526,
       -1.12513744,  0.05898229, -0.13332464,  1.18813757, -2.64642863,
       -2.04551525, -0.80148949,  0.34418139, -0.68623293, -0.82034803,
       -2.12425773, -0.48737345, -0.72966661, -0.54491771, -0.77889521,
       -1.74744345, -0.94360517, -0.6551269 , -0.14836124, -0.95607365,
       -1.24917671, -1.05395059, -0.69084956, -0.79875683, -0.4718144 ,
        0.17962822, -0.72267543, -0.23531312, -0.76383685, -0.45872792,
       -1.77504914, -0.8707672 , -0.76806047, -0.90452459, -0.12876294])
```

The length of Y is 100. $\beta_0 = -1, \beta_1 = 0.5$

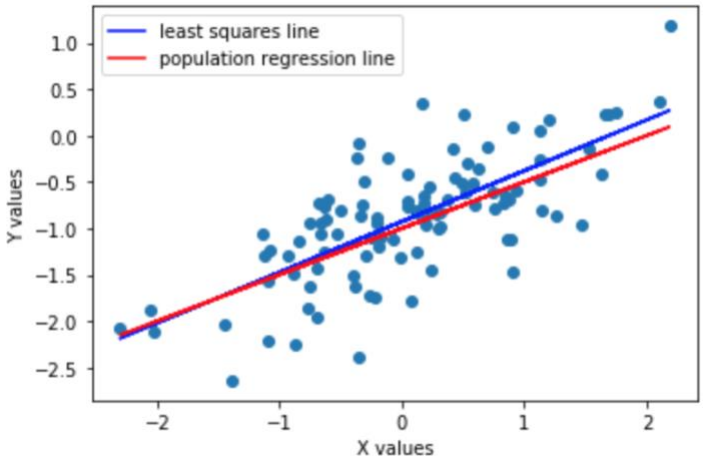
d) Below is the scatter plot of Y-values against x-values.



There is a positive linear relationship between Y and x. The value of Y increases linearly as x value increases.

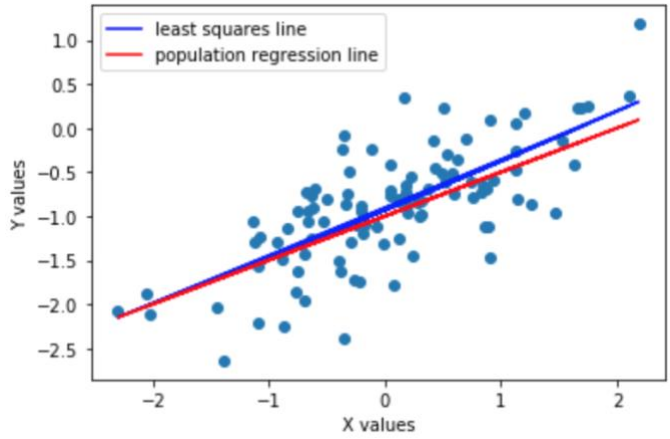
e) Given the p-value = 2.20e-17, very close to 0, the linear model is significant.
With coefficients $\hat{\beta}_0 = -0.9265 \; and \; \hat{\beta}_1 = 0.5477$. Both are very close to the true $\beta_0 \; and \; \beta_1$.

f) Below is the plot with least squares line and population regression line.

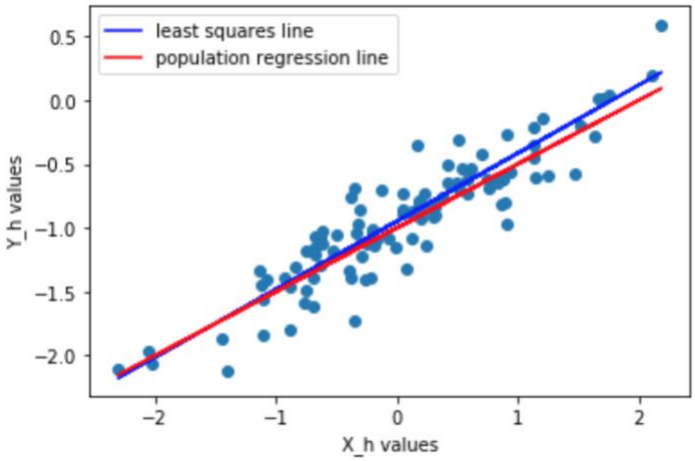


Very closely overlapped, but get further apart a little bit as x-value increases.

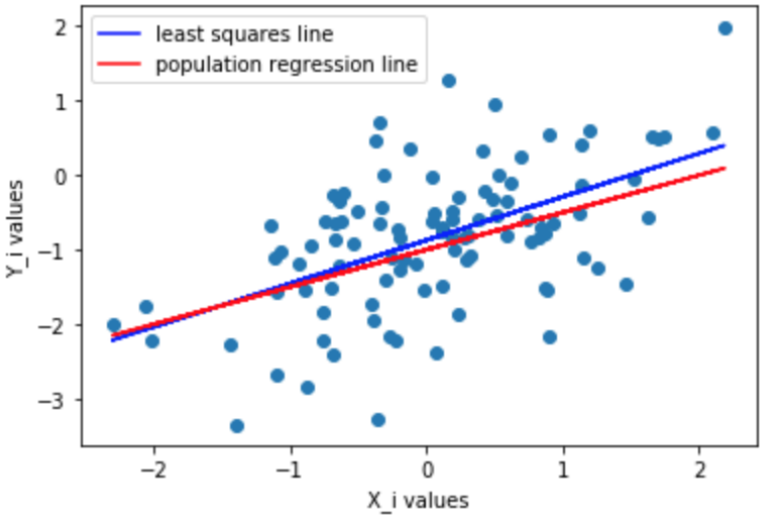
g) The p-value for the new linear model is still very small.
But the p-value for added quadratic term x^2 is very large, making it an insignificant predictor. Therefore, there is no linear relationship between x^2 and y.
Also, below is the comparison of least squares line and population regression line for the new linear model. It is basically the same compared to the plot in part (f). Therefore, there is not enough evidence that the quadratic term improves the model fit.



h) Decreasing the variance of 'eps' to 0.05:
The coefficients $\hat{\beta}_0 = -0.9671 \; and \; \hat{\beta}_1 = 0.5213$ are still very close to $\beta_0 = -1 \; and \; \beta_1 = 0.5$. But R^2 grows from 0.522 to 0.832 and Standard Errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ decrease by over 50%. Also, the scatter plot below shows a stronger linear relationship between x and Y values, and two lines overlap a lot with less noise. Obviously, the model fits the data much better.



i) Increasing the variance of 'eps' to 0.75:
The coefficients $\hat{\beta}_0 = -0.8727 \; and \; \hat{\beta}_1 = 0.5826$ are further away from $\beta_0 = -1 \; and \; \beta_1 = 0.5$. Moreover, R^2 decreases from 0.522 to 0.292 and Standard Errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ increase to 0.081 and 0.092. Also, the scatter plot below shows a weaker linear relationship between x and Y values, and two lines are further away from each other with more noise. The model here fits the data much worse.



j) Confidence Intervals:
Original Dataset:
$\hat{\beta}_0$: [-1.01974096 -0.83324551] $\hat{\beta}_1$: [0.44261338 0.65281376]

Less Noisy Dataset:
$\hat{\beta}_0$: [-1.00882843 -0.92542512] $\hat{\beta}_1$: [0.47433592 0.56834039]

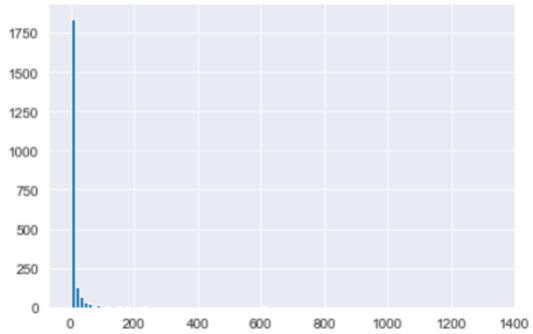
Noisier Dataset:
$\hat{\beta}_0$: [-1.03419235 -0.71117274] $\hat{\beta}_1$: [0.40060345 0.76468119]

Question 4

1. Most of the variables in the dataset are categorical variables. After plotting the histograms of 3 continuous variables without missing values to check the distributions, none of the 3 continuous variables above follow the Gaussian Distribution. While the ranges are all very large, most values falls into a very small range on the left side. They are all too skewed to the same direction. Therefore, they are not quite useful when used as predictors in Naive Bayes classifier, using the Gaussian NB from 'sklearn' for these 3 continuous features.

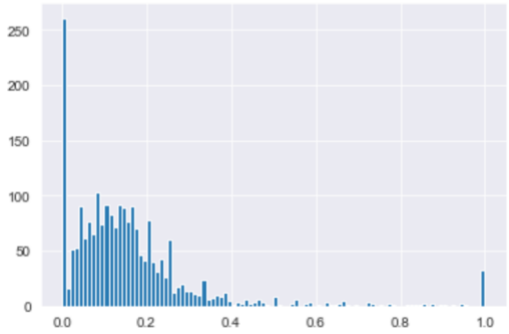
```
1 spam['size.kb'].hist(bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
```



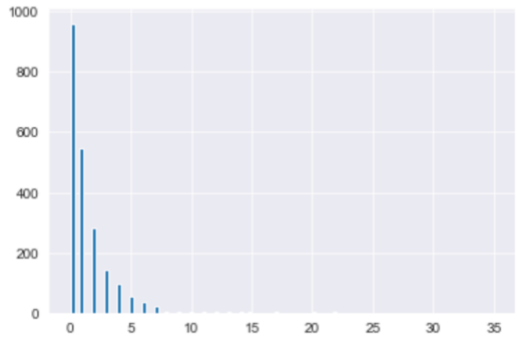

```
1 spam['cappct'].hist(bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7
```



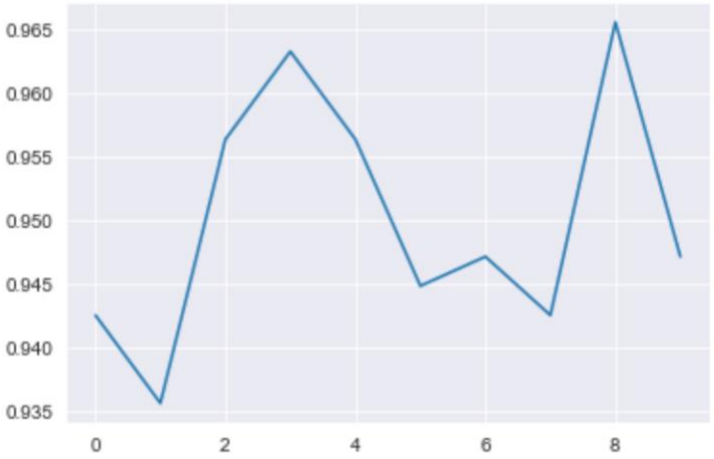

```
1 spam['special'].hist(bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7
```



Continue to only use only the categorical variables as features to train the classifier on 80% training set. The accuracy on the 20% testing set is 93.79%, which is pretty high. Therefore, the generalization error is about 6.21%.

2. Repeat 10 times. Plot the variability trend.



The accuracy goes up and down. The MIN is 0.9356 and the MAX is 0.9655. In conclusion, the accuracy varies a little bit, but always stays above 94%, which is pretty high.

3. The generalization error (test error) should be treated more seriously when it is low, compared to the train error. The train error can be less important, if test error stays at a very low level. We can implement cross validation to select fewer but more important features to improve the accuracy of the classifier.

Question 5 MLE

From least squares estimation:

$$\hat{\beta}_{least\_squares} = (X^T X)^{-1} X^T y$$

Now prove:

$$\hat{\beta}_{MLE} = \hat{\beta}_{least\_squares}$$

Given the sample $\{(X_i, Y_i)\}_{i=1}^n$ and $Y = X^T\beta + \varepsilon, \varepsilon \sim N(0, \sigma^2)$,

we have: $Y_i|(X_{i1} = x_{i1}, \dots, X_{ik} = x_{ik}) \sim N(\beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik}, \sigma^2)$.

In matrix form, it will be: $Y|X \sim N_n(X\beta, \sigma^2 I)$.

Compute the log-likelihood function of Y conditional on $(X_1, \dots, X_n)$:

$$log\ likelihood(\beta) = \log_e(Y; X\beta, \sigma^2 I) = \sum_{i=1}^{n} \log_e(Y_i; (X\beta)_i, \sigma$$

$$= -\log\left((2\pi)^{\frac{n}{2}}\sigma^n\right) - \frac{1}{2\sigma^2}(Y - X\beta)^T(Y - X\beta)$$

Set the derivative of the log likelihood function with respect to β equal to 0 to get $\hat{\beta}_{MLE}$.

$$\frac{\partial AX}{\partial X} = A = 0$$

$$\frac{1}{\sigma^2}(Y - X\beta)'X = \frac{1}{\sigma^2}(Y'X - \beta'X'X) = 0$$

Result:

$$\hat{\beta} = \hat{\beta}_{MLE} = (X^T X)^{-1} XY = \hat{\beta}_{least\ squares}$$

Question 6 KNN

a) Given X = 0.6, we use observations in the range [0.55, 0.65],
we are using the interval [x-0.05, x+0.05] when 0.05<=x<=0.95, using [0, x+0.05] when x<0.05, and using [x-0.05, 1] when x>0.95.
To compute the average fraction:

$$\int_{0.05}^{0.95} 10dx + \int_{0}^{0.05} (100x + 5)dx + \int_{0.95}^{1} (105 - 100x)dx = 9.75$$

b) p= 2 features, assuming $X_1\ and\ X_2$ are independent, the square of the fraction in part (a) will be about 0.951%

c) Similar to (b), when p = 100 features, since the observations are uniformly distributed on each feature and the values all $\in [0,1]$, the fraction of the available observations will be $9.75\%^{100} \approx 0$.

d) The drawback of KNN when p is large is that the fraction of available observations we will use to make predictions will be very close to 0. This makes the algorithm basically useless, as we don't have enough points.

e) p = 1, l = 0.1
p = 2, l = $0.1^{\frac{1}{2}}$
p = 100, l = $0.1^{\frac{1}{100}}$

As p increases, l decreases very quickly. It means for the 10% of the training observations to be used for prediction, the ‘hypercube’ must extend further and further in the same pace l decreases. As a result, the selected 10% observations aren’t really close to each on the surface of the ‘hypercube’, which is the opposite we want for KNN.