Problem 1. Suppose we fit n data points with a line by minimizing RSS (least squares), and that we want to estimate the line at a new point, x_0. Denoting its value on the line by μ_0, the estimate is:

$$\hat{\mu}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0$$

(a) Find variance of $\hat{\mu}_0$.

(b) The standard deviation of $\hat{\mu}_0$ can be expressed as a function of $(x_0 - \bar{x})$. Find this function and briefly explain its shape.

(c) Find 95% confidence interval for $\mu_0 = \beta_0 + \beta_1 x_0$ under assumption of normality. (n is large enough)

(a) To find $\hat{\beta}_0$ and $\hat{\beta}_1$: $\begin{cases} \frac{d}{d\hat{\beta}_0} RSS(\hat{\beta}_0, \hat{\beta}_1) = 0 \\ \frac{d}{d\hat{\beta}_1} RSS(\hat{\beta}_0, \hat{\beta}_1) = 0 \end{cases} \Rightarrow \begin{cases} \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \\ \hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \end{cases}$

$Var(\hat{\mu}_0) = Var(\hat{\beta}_0 + \hat{\beta}_1 x_0) = Var(\bar{y} - \hat{\beta}_1 \bar{x} + \hat{\beta}_1 x_0)$

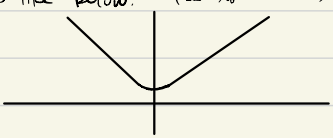
$= Var(\bar{y}) + Var(\hat{\beta}_1) \cdot (x_0 - \bar{x})^2$

$Var(\bar{y}) = \frac{\sigma^2}{n}$ $\quad Var(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum_{i=1}^n x_i)^2}$

$= \frac{\sigma^2}{n} + (x_0 - \bar{x})^2 \frac{\sigma^2}{\sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum_{i=1}^n x_i)^2}$

(b) standard deviation of $\hat{\mu}_0$ is $s = \sqrt{\frac{\sigma^2}{n} + (x_0 - \bar{x})^2 \frac{\sigma^2}{\sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum_{i=1}^n x_i)^2}}$

it's shape looks like below: (as x_0 increases, first decrease then increase)



(c) $\alpha = 0.05$, $Z_{\alpha/2} = Z_{0.025} = 1.96$

for $Pr\{\mu \in [\hat{\mu}_0 - \beta, \hat{\mu}_0 + \beta]\}$, $\beta = s \cdot Z_{0.025} = 1.96s$, where s is from (b)

$\therefore$ 95% CI:

$$\left[\hat{\mu}_0 - 1.96\sqrt{\frac{\sigma^2}{n} + (x_0 - \bar{x})^2 \frac{\sigma^2}{\sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum_{i=1}^n x_i)^2}},\ \hat{\mu}_0 + 1.96\sqrt{\frac{\sigma^2}{n} + (x_0 - \bar{x})^2 \frac{\sigma^2}{\sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum_{i=1}^n x_i)^2}}\right]$$

Problem 2. Assume that $X \sim N(0,1)$, $E \sim N(0,1)$, X and E are independent, and $Y = X + \beta E$. Show that:

$$r_{XY} = \frac{1}{\sqrt{\beta^2+1}} \qquad \frac{Cov(X,Y)}{1\times\sqrt{1+\beta^2}} = \frac{1}{\sqrt{\beta^2+1}}$$

Note that r_{XY} is defined as $r_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$.

$\because X\sim N(0,1)$, $E\sim N(0,1)$ $\therefore Var(X) = Var(E) = 1$ $\therefore \sigma_X = 1$

$Var(Y) = Var(X+\beta E) = Var(X) + \beta^2 Var(E) = 1+\beta^2$ $\therefore \sigma_Y = \sqrt{1+\beta^2}$

$Var(Y-X) = Var(Y) + Var(X) + 2\times 1\times(-1)\, Cov(X,Y) = Var(\beta E)$

$1+\beta^2 + 1 - 2Cov(X,Y) = \beta^2 Var(E)$

$2+\beta^2 - 2Cov(X,Y) = \beta^2$

$Cov(X,Y) = 1$

$\therefore r_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{1}{1\times\sqrt{1+\beta^2}} = \frac{1}{\sqrt{\beta^2+1}}$

Problem 3. Suppose there are n data points $\{x_i \in \mathbb{R}, y_i \in \mathbb{R}\}_{i=1}^n$. We fit a line $y = a + bx$ with least squares, and another line $x = c + dy$ with least squares. Show that $bd \leq 1$, and briefly explain when $bd = 1$ and what it means.
Hint: Cauchy-Schwarz Inequality. $|\mathrm{Cov}(X,Y)|^2 \leq \mathrm{Var}(X)\cdot\mathrm{Var}(Y)$.

$Var(X)Var(Y) = Var(c+dy)\, Var(a+bx)$

$= b^2 d^2 Var(x) Var(y)$

using Cauchy-Schwarz:

$|Cov(X,Y)|^2 \leq Var(X)\cdot Var(Y)$

$|Cov(X,Y)|^2 \leq b^2 d^2 Var(x) Var(y)$

$|Cov(X,Y)| \leq bd\, \sigma_X \sigma_Y$

Since $Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$, $|Corr(X,Y)| \leq 1$

$\therefore \frac{|Cov(X,Y)|}{\sigma_X \sigma_Y} \leq bd$

$|Corr(X,Y)| \leq bd$

$bd \leq 1$

if $bd = 1$, x and y are perfectly correlated, they are linearly dependent

Problem 4. (Extra Credit) A student wants to predict a variable, $Y \in \mathbb{R}^n$, from two other variables, $X_1 \in \mathbb{R}^n$ and $X_2 \in \mathbb{R}^n$, using multiple regression. The student defines a new variable $X_3 = X_1 + X_2$ and uses multiple regression to predict Y from X_1, X_2, X_3. Show why this method is problematic.
Hint 1: $A_{n\times n}$ is invertible $\Leftrightarrow$ $\text{Rank}(A) = n$.
Hint 2: $\text{Rank}(AB) \leq \min(\text{Rank}(A), \text{Rank}(B))$.

$n \{ [\ X_1\ ]$ $\quad n [\ X_{i2}\ 3\ ]_{n\times 3}$ $\quad X^{-1}X \Rightarrow 3\times 3$, rank $= 2$
$\quad 3\times n \quad n\times 3$

After adding the new variable, given $X_1 \in \mathbb{R}^n$, $X_2 \in \mathbb{R}^n$
the shape of matrix X is $n\times 3$, the shape of X^TX is 3×3
Given $X_3 = X_1 + X_2$, the rank of $X^TX \leq 2$ (hint 2)
denote $A = X^TX$, $\text{Rank}(A_{3\times 3}) \leq 2$
Hint 1 $\Rightarrow$ X^TX is singular or non-invertible.
this is because its columns are not all linearly independent
So when the student solve for the solution $\vec{\beta} = (X^TX)^{-1}X^TY$ (*)
there are either no solution or infinitely many solutions
thus the equation (*) is undetermined, this method is problematic.

Problem 5: Simple Linear Regression

In this question, you will implement simple linear regression from scratch. The dataset you will work with is called the Boston data set. You can find more information about the data set here: https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

You will use the pandas library to load the csv file into a dataframe:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# read the csv file and load into a pandas dataframe
# make sure Boston.csv is in the same file path as this notebook
boston = pd.read_csv('Boston.csv')
```

In [3]:
```
# read the above link to learn more about what each of the columns indicate
boston.head()
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Simple linear regression builds a linear relationship between an input variable X and an output variable Y. We can define this linear relationship as follows:

$$Y = \beta_0 + \beta_1 X$$

Objective: find the linear relationship between the proportion of non-retail business acres per town (indus) and the full-value property-tax rate per 10,000 dollars (tax)

So our equation will look like:

$$TAX = \beta_0 + \beta_1 INDUS$$

Here, the coefficient β_0 is the intercept, and β_1 is the scale factor or slope. How do we determine the values of these coefficients?

There are several different methods to do so, but we will focus on the Ordinary Least Squares (OLS) method. This method minimizes the sum of the squares of the differences between the observed dependent variable and those predicted by the linear function.

Recall that a residual is the difference between any data point and the line of regression. When we develop a regression model, we want the sum of the residuals squared to be minimized, indicating that the model is a close fit to the data.

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

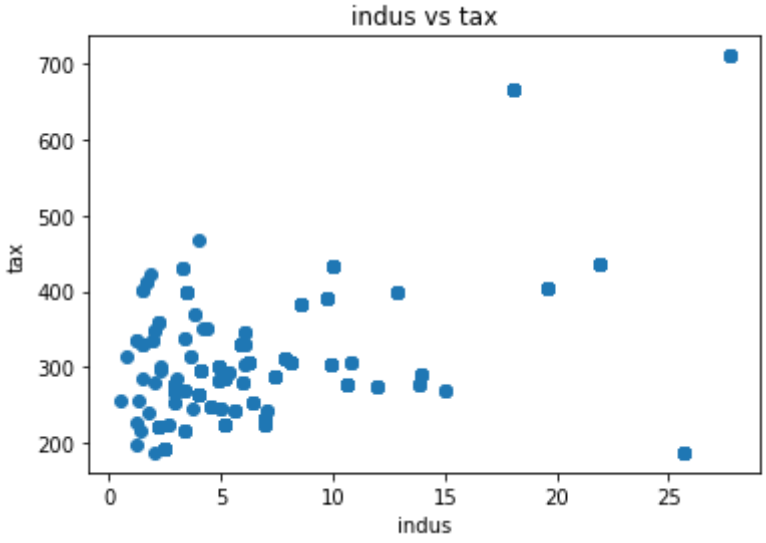
$$= \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$

This is the objective function we minimize to find β_0 and β_1.

In [4]:
```
# set X to 'indus' and y to 'tax'
X = boston['indus']
y = boston['tax']
```

First, visualize the data by plotting X and y using matplotlib. Be sure to include a title and axis labels.

In [8]:
```
# TODO: display plot
plt.scatter(X,y)
# TODO: labels and title
plt.xlabel('indus')
plt.ylabel('tax')
plt.title('indus vs tax')
plt.show()
```



TODO: What do you notice about the relationship between the variables?

A: Generally, the higher then indus, the higher the tax. The points are mostly distributed in a certain range (left bottom), with a few outlier points distributed on the right side.

Next, find the coefficients. The values for β_0 and β_1 are given by the following equations, where n is the total number of values. This derivation was done in class.

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1\bar{x}$$

In [8]:
```
# TODO: implement function
def get_coeffs(X, y):
    '''
    Params:
        X: the X vector
        y: the y vector
    Returns:
        a tuple (b1, b0)
    '''
    X,y = np.array(X), np.array(y)
    X_bar = np.mean(X)
    y_bar = np.mean(y)
    beta_1_numerator = sum([(X[i]-X_bar)*(y[i]-y_bar) for i in range(len(y))])
    beta_1_denominator = sum([(i-X_bar)**2 for i in X])
    beta_1 = beta_1_numerator/beta_1_denominator
    beta_0 = y_bar - beta_1*X_bar
    # x = np.hstack((np.ones((X.shape[0], 1)), X))
    # W = np.linalg.inv(x.T @ x) @ x.T @ y
    # w, b = W[1:, :], W[0, 0]
    return beta_0, beta_1
    raise NotImplementedError
# print(X.shape)
# get_coeffs(X, y)
```

In [9]:
```
# run cell to call function and display the regression line
# the values are rounded for display convenience
b0, b1 = get_coeffs(X, y)
print("Regression line: TAX = " + str(round(b0)) + " + " + str(round(b1)) +"*IND
```

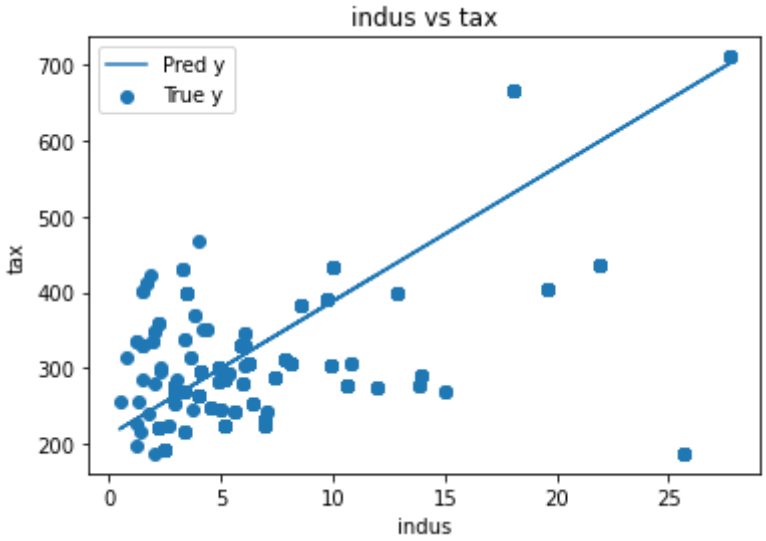
```
Regression line: TAX = 211 + 18*INDUS
```

Plot the regression line overlayed on the real y-values.

In [10]:
```
# TODO: plot y-values
plt.figure()
plt.scatter(X,y,label = 'True y')

# TODO: plot regression line
y_pred = [b0 + b1*x for x in X]
plt.plot(X,y_pred, label = 'Pred y')

# TODO: labels and title
plt.xlabel('indus')
plt.ylabel('tax')
plt.title('indus vs tax')
plt.legend()
plt.show()
```



The line appears to fit the data, but first, let us find the RSS to evaluate this model. The RSS is used to measure the amount of variance in the data set that is not explained by the regression model. Recall that

$$RSS = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$

In [11]:
```
# TODO: implement function
def get_RSS(b0, b1, X, y):
    '''
    Params:
        b0: beta 0
        b1: beta 1
        X: X vector
        y: y vector
    Returns:
        residual sum of squares (RSS)
    '''
    rss = 0
    X,y = np.array(X), np.array(y)
    for i in range(len(y)):
        rss += (y[i]- (b0 + b1*X[i]) )**2
    return rss
    raise NotImplementedError
# print(type(X),type(y))
```

In [12]:
```
# run this cell to print RSS
print("RSS:", get_RSS(b0, b1, X, y))
```

```
RSS: 6892554.224031534
```

We can also evaluate the model through the Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2 score).

- The RMSE is similar to the RSS, but provides a value with more interpretable units -- in our case, tax rate per 10,000 dollars.

- The R^2 value represents the proportion of the variance for the dependent variable that is explained by the independent variable.

Use the following equations to find the RMSE and R^2 score:

$$RMSE = \sqrt{(\sum_{i=1}^{n} \frac{1}{n}(\hat{y}_i - y_i)^2)}$$

$R^2 = 1 - \frac{SS_r}{SS_t}$ where

$$SS_t = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

and

$$SS_r = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In [13]:
```
# TODO: implement function
# import math
def get_RMSE(b0, b1, X, y):
    '''
    Params:
        b0: beta 0
        b1: beta 1
        X: X vectore
        y: y vector
    Returns:
        rmse
    '''
    X,y = np.array(X), np.array(y)
    n = len(y)
    total = sum([1/n * ( (b0+b1*X[i]) - y[i])**2 for i in range(n)])
    return np.sqrt(total)
    raise NotImplementedError
```

In [14]:
```
# run cell to print RMSE
print("RMSE: ", get_RMSE(b0, b1, X, y))
```

```
RMSE:  116.71181887064354
```

In [15]:
```
# TODO: implement function
def get_R2(b0, b1, X, y):
    '''
    Params:
        b0: beta 0
        b1: beta 1
        X: X vector
        y: y vector
    Returns:
        r2 score
    '''
    X,y = np.array(X), np.array(y)
```

```
    y_bar = np.mean(y)
    SSt = sum([(y[i]-y_bar)**2 for i in range(len(y))])
    SSr = sum([(y[i] - (b0 + b1*X[i]))**2 for i in range(len(y))])
    return 1 - SSr / SSt
    raise NotImplementedError
```

In [16]:

```
# run cell to print RMSE
print("R2: ", get_R2(b0, b1, X, y))
```

```
R2:  0.5194952370037822
```

TODO: Analyze what the above R^2 score indicates about the model.

A: It means that about 52% variation of the tax can be explained by indus.

Now, we will compare the above results with the results from using scikit-learn, a machine learning library in Python. Read the documentation (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) to learn how to use this library. Return the R^2 score and RMSE.

In [21]:

```
# TODO: scikit learn function
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

def linear_regression_SKL(X, y):
    '''
    Params:
        X: X vector
        y: y vector
    Returns:
        rmse and r2 as a tuple
    '''
    X,y = np.array(X), np.array(y)
    clf = LinearRegression()
    clf.fit(X.reshape(-1, 1),y)
    y_pred = clf.predict(X.reshape(-1, 1))
    b1 = clf.coef_[0]
    b0 = clf.intercept_
    # rmse = get_RMSE(b0, b1, X, y)
    # print(rmse)
    rmse = np.sqrt(mean_squared_error(y,y_pred))
    # r2 = get_R2(b0, b1, X, y)
    r2 = clf.score(X.reshape(-1, 1),y)
    return (rmse,r2)
    raise NotImplementedError
# linear_regression_SKL(X, y)
```

Out[21]:

```
(116.71181887064392, 0.5194952370037791)
```

In [22]:

```
# run this cell to print results from SKL LR
linear_regression_SKL(X, y)
```

Out[22]:

```
(116.71181887064392, 0.5194952370037791)
```

TODO: Analyze the results and compare the RMSE and R^2 to the previous method.

A: From the results, the standard deviation of the residuals (error of predicted tax value) is 116.7, and the 52% (R^2) variation of the tax can be explained by indus. The RMSE and R^2 are the same as the previous method.