Final Project

In this notebook,first we will studty a regression model and the we will look at a classification case.

Regression

We will build a model to find the relation between the wind speed and spread area for a forest fire case. The dataset gives data of the valueas of different features like temperature, FFMC,DMC,ISI,temp,wind(in km/h),area(in he) etc.. but we will only use the ISI and temp for this model.

ISI(Initial spread index)- It is a numeric rating of the expected rate of fire spread. temp-temperature in Celsius Degree.

In [154]:
```
import csv
f=open("forestfires.csv",'r')
header=f.readline().strip().split(',')
header
```

Out[154]:
```
['X',
 'Y',
 'month',
 'day',
 'FFMC',
 'DMC',
 'DC',
 'ISI',
 'temp',
 'RH',
```

```
 'wind',
 'rain',
 'area']
```

we see that the wind feature has an index 10 and the area feature has index 12.

In [155]:
```
dataset=[]
for line in f:
    line=line.split(',')
    dataset.append(line)
dataset[155]
```

Out[155]:
```
['7',
 '4',
 'aug',
 'sun',
 '94.8',
 '108.3',
 '647.1',
 '17',
 '16.4',
 '47',
 '1.3',
 '0',
 '1.56\n']
```

now we have all cases as a list inside a list dataset

In [156]:
```
lab=[float(d[7]) for d in dataset]
len(lab)
```

Out[156]:
```
517
```

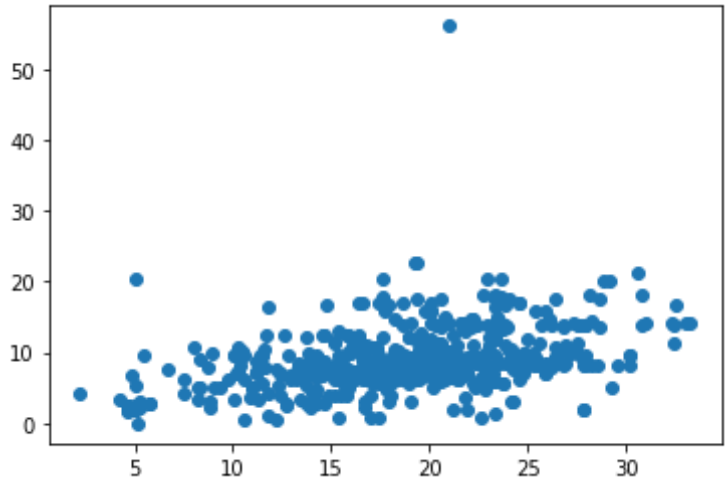
In [157]:
```
feat=[float(d[8]) for d in dataset]
len(feat)
```

Out[157]:
```
517
```

First lets get a visual representation of how the data looks.

In [158]:
```
import matplotlib.pyplot as plt
plt.scatter(feat,lab)
```

Out[158]:
```
<matplotlib.collections.PathCollection at 0xdbedb2c508>
```



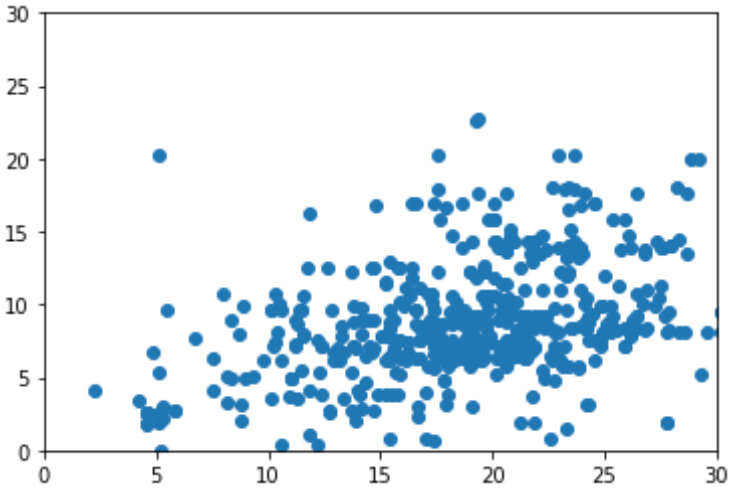
We see that the ISIvalue increases with the increase in temperature, but with a very small slope.
And also lets get rid of the outliers, those whose ISI is greater than 30

In [159]:
```
F_lab=[float(d[7]) for d in dataset if float(d[7])<30]
F_feat=[float(d[8]) for d in dataset if float(d[7])<30]
print(len(F_lab),len(F_feat))
```

```
516 516
```

In [160]:
```
plt.ylim(0,30)
plt.xlim(0,30)
plt.scatter(F_feat,F_lab)
```

Out[160]:
```
<matplotlib.collections.PathCollection at 0xdbed9e5508>
```



Now that we have got rid of the outlier let's find the best fit. Lets create the label y vector and the feature X matrix

In [161]:
```
y=F_lab                                                    # The la
bel vector
```

In [162]:
```
X=[[1, d] for d in F_feat]                                 # the fea
ture matrix
```

Next we find the values of theta, we will have to import the numpy module for this

In [163]:
```
import numpy
```

In [ ]:
```
theta,residual,rank,S=numpy.linalg.lstsq(X,y)
```

In [166]:
```
theta
```

Out[166]:
```
array([3.19017284, 0.30395711])
```

Now that we have the value of theta we can predict the unknown value using the formula.
theta1 + X*theta2

In [ ]:

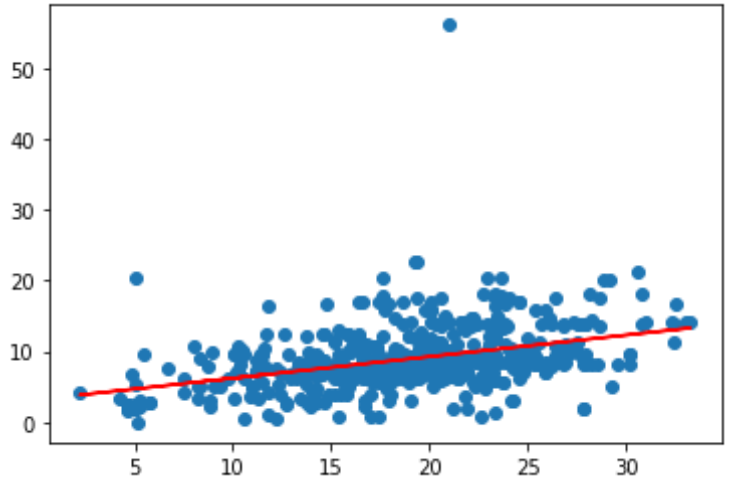
Lets see the best fit line in the scatter plot

In [167]:
```
def predictedvalue(d):
    return sum(d*theta)
```

In [168]:
```
y_predicted=[predictedvalue(d) for d in X]
```

In [169]:
```
plt.scatter(feat,lab)
plt.plot(F_feat,y_predicted,color='red')
```

Out[169]:
```
[<matplotlib.lines.Line2D at 0xdbefee3a48>]
```



The above red line gives us the best fit linear line for the given data

the above defined function (predictedvalue) predicts the value of y when and list d is given to it

here the items in d are 1, temp

Classification

Now we builda classification model.

The Banknote Dataset involves predicting whether a given banknote is authentic given a number of measures taken from a photograph.

It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 1,372 observations with 4 input variables and 1 output variable. The variable names are as follows:

Variance of Wavelet Transformed image (continuous). Skewness of Wavelet Transformed image (continuous). Kurtosis of Wavelet Transformed image (continuous). Entropy of image (continuous). Class (0 for authentic, 1 for inauthentic).

In [ ]:
```
```

In [2]:
```
f=open('data_banknote_authentication.txt', 'r')
```

In [3]:
```
data=f.readlines()
```

In [4]:
```
dataset=[]
for l in data:
    dataset.append(l)
len(dataset)
```

Out[4]:
```
1372
```

In [5]:
```
dataset[:8]
```

Out[5]:
```
['3.6216,8.6661,-2.8073,-0.44699,0\n',
 '4.5459,8.1674,-2.4586,-1.4621,0\n',
 '3.866,-2.6383,1.9242,0.10645,0\n',
 '3.4566,9.5228,-4.0112,-3.5944,0\n',
 '0.32924,-4.4552,4.5718,-0.9888,0\n',
 '4.3684,9.6718,-3.9606,-3.1625,0\n',
 '3.5912,3.0129,0.72888,0.56421,0\n',
 '2.0922,-6.81,8.4636,-0.60216,0\n']
```

We see that the above dataset is not random arranged, all the data that have authentic note are arranged first. So first we have to randomly shuffle the data.

In [6]:
```
import random
```

In [66]:
```
random.shuffle(dataset)
```

In [67]:
```
dataset[:10]
```

Out[67]:
```
['-1.7344,2.0175,7.7618,0.93532,0\n',
 '1.296,4.2855,-4.8457,-2.9013,1\n',
 '3.3397,-4.6145,3.9823,-0.23751,0\n',
 '-2.659,-1.6058,1.3647,0.16464,1\n',
 '4.2478,7.6956,-2.7696,-1.0767,0\n',
 '0.040498,8.5234,1.4461,-3.9306,0\n',
 '-2.0759,10.8223,2.6439,-4.837,0\n',
 '1.0194,1.1029,-2.3,0.59395,1\n',
 '-5.8818,7.6584,0.5558,-2.9155,1\n',
 '0.3292,-4.4552,4.5718,-0.9888,0\n']
```

Now we shall split the data to train set and traning set. We will split the data by 3:1 ratio(train data 75% and test data 25%).
The total dataset has 1372 data point we will use 3*343=1029 for traning and 343 for test

In [80]:
```
dataset_train=dataset[:1029]
dataset_test=dataset[1029:]
print(len(dataset_train),len(dataset_test))
```

```
1029 343
```

In [81]:
```
dataset_train[5],dataset_test[5]
```

Out[81]:
```
('0.040498,8.5234,1.4461,-3.9306,0\n',
 '-2.6864,-0.097265,0.61663,0.061192,1\n')
```

Now that we have split the dataset let's seperate the dependent and independent variables

In [132]:
```
def indepfeature(d):
    l=[float(i) for i in d.split(',')[:-1]]
    return l
def depfeature(d):
    l=int(d.split(',')[-1][0])
    return l
```

In [133]:
```
depfeature(dataset_train[0])
```

Out[133]:
```
0
```

In [134]:
```
X_train=[indepfeature(d) for d in dataset_train]
X_test=[indepfeature(d) for d in dataset_test]
y_train=[depfeature(d) for d in dataset_train]
y_test=[depfeature(d) for d in dataset_test]
```

In [136]:
```
X_train[0:3],X_test[0:3],y_train[0:3],y_test[0:3]                # chec
k if it matches with the above dataset before spliting in line
```

Out[136]:
```
([[-1.7344, 2.0175, 7.7618, 0.93532],
  [1.296, 4.2855, -4.8457, -2.9013],
  [3.3397, -4.6145, 3.9823, -0.23751]],
 [[-2.7908, -5.7133, 5.953, 0.45946],
  [-1.6988, -7.1163, 5.7902, 0.16723],
  [4.0524, 5.6802, -1.9693, 0.026279]],
 [0, 1, 0],
 [1, 1, 0])
```

We have split our dataset as required.

Now let's import the KNN algo from sklearn for the classification. We shall use a range of K values to find the best fit.

In [137]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
model=classifier.fit(X_train, y_train)
```

In [146]:
```
y_predict=model.predict(X_test)
```

In [149]:
```
correctprediction = y_predict==y_test
```

In [150]:
```
correctprediction
```

Out[150]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
```

```
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True])
```

Guys I don't kmow if it is a magic or if there is a bug. But this model is 100% accurate

But let's check it again with a datapoint

In [153]:
```
model.predict([[-2.7908, -5.7133, 5.953, 0.45946],
  [-1.6988, -7.1163, 5.7902, 0.16723],
  [4.0524, 5.6802, -1.9693, 0.026279]])
```

Out[153]:
```
array([1, 1, 0])
```

check it out[136] in line[136]

In [ ]:
```

```