

INF -552 (Machine Learning for Data Informatics)
Homework1 – Report

Forest Fire Data

In this exercise, we investigate a difficult regression task, where the aim is to predict the burned area of forest fires in the northeast region of Portugal.

(a) Download the Forest Fire data from: <https://archive.ics.uci.edu/ml/datasets/Forest+Fires>.

(b) Exploring the data:

i. How many rows are in this data set? How many columns? What do the rows and columns represent?

The data set consists of 517 rows and 13 columns.

Each row represents a training data .Hence we have 517 training samples.

Each column represents a feature/predictor except for the last column which is the dependent variable or output.

Description of the column variables as provided with the dataset.

1. X - x-axis spatial coordinate within the Montesinho park map: 1 to 9
2. Y - y-axis spatial coordinate within the Montesinho park map: 2 to 9
3. month - month of the year: 'jan' to 'dec'
4. day - day of the week: 'mon' to 'sun'
5. FFMC - FFMC index from the FWI system: 18.7 to 96.20
6. DMC - DMC index from the FWI system: 1.1 to 291.3
7. DC - DC index from the FWI system: 7.9 to 860.6
8. ISI - ISI index from the FWI system: 0.0 to 56.10
9. temp - temperature in Celsius degrees: 2.2 to 33.30
10. RH - relative humidity in %: 15.0 to 100
11. wind - wind speed in km/h: 0.40 to 9.40
12. rain - outside rain in mm/m² : 0.0 to 6.4
13. area - the burned area of the forest (in ha): 0.00 to 1090.84

ii. Explain why the transformation $Y_1 = \ln(1 + Y)$, where Y is the response variable is useful for this dataset. In the following, use Y_1 as the new response variable.

The area data seems to be skewed towards the value 0. hence $\ln(1+Y)$ would reduce the skewness of the data and henceforth create a healthy distribution for regression analysis.

iii. Make pairwise scatterplots of the predictors (columns) in this data set with the dependent variable. Describe your findings.

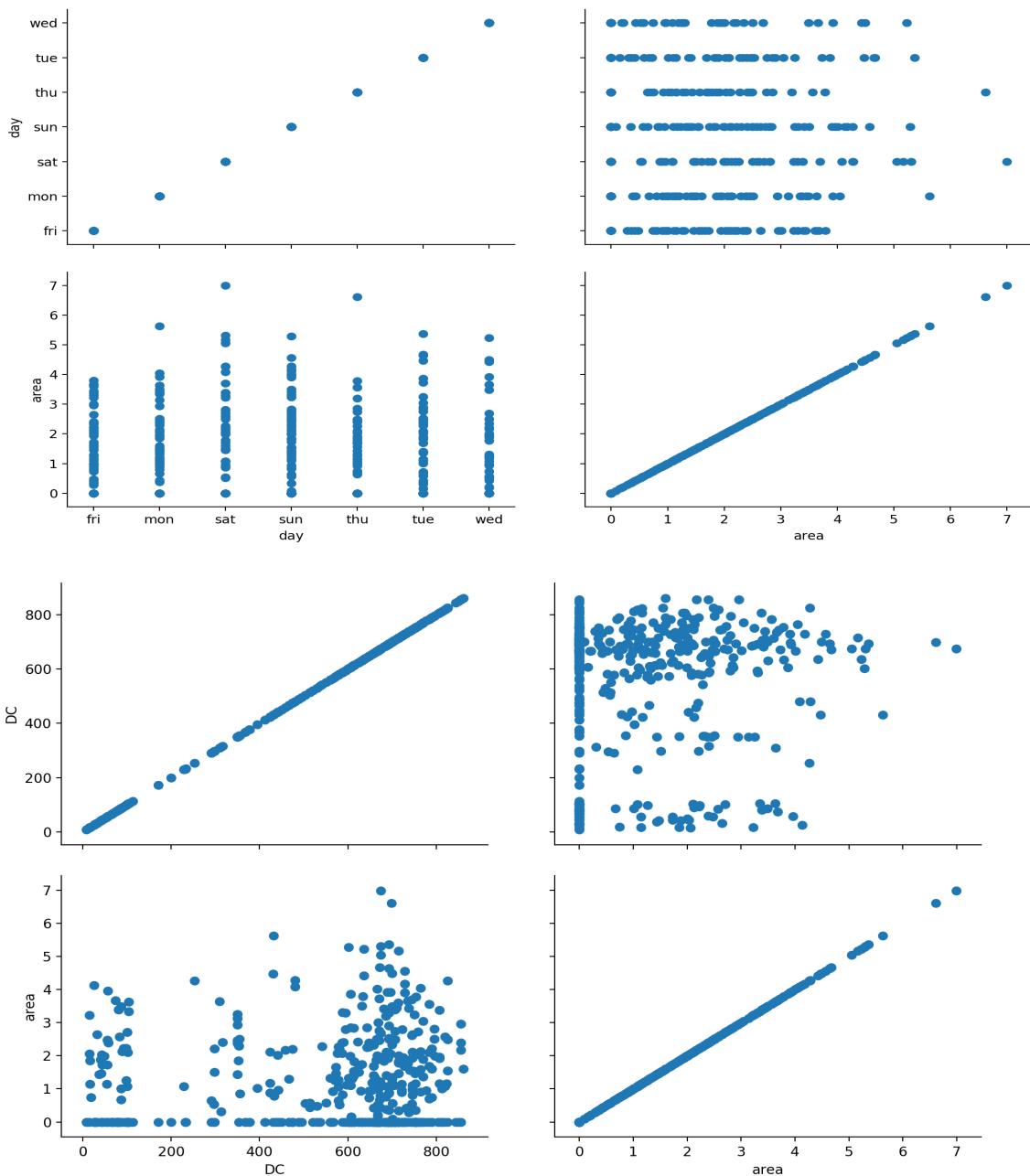
Code:

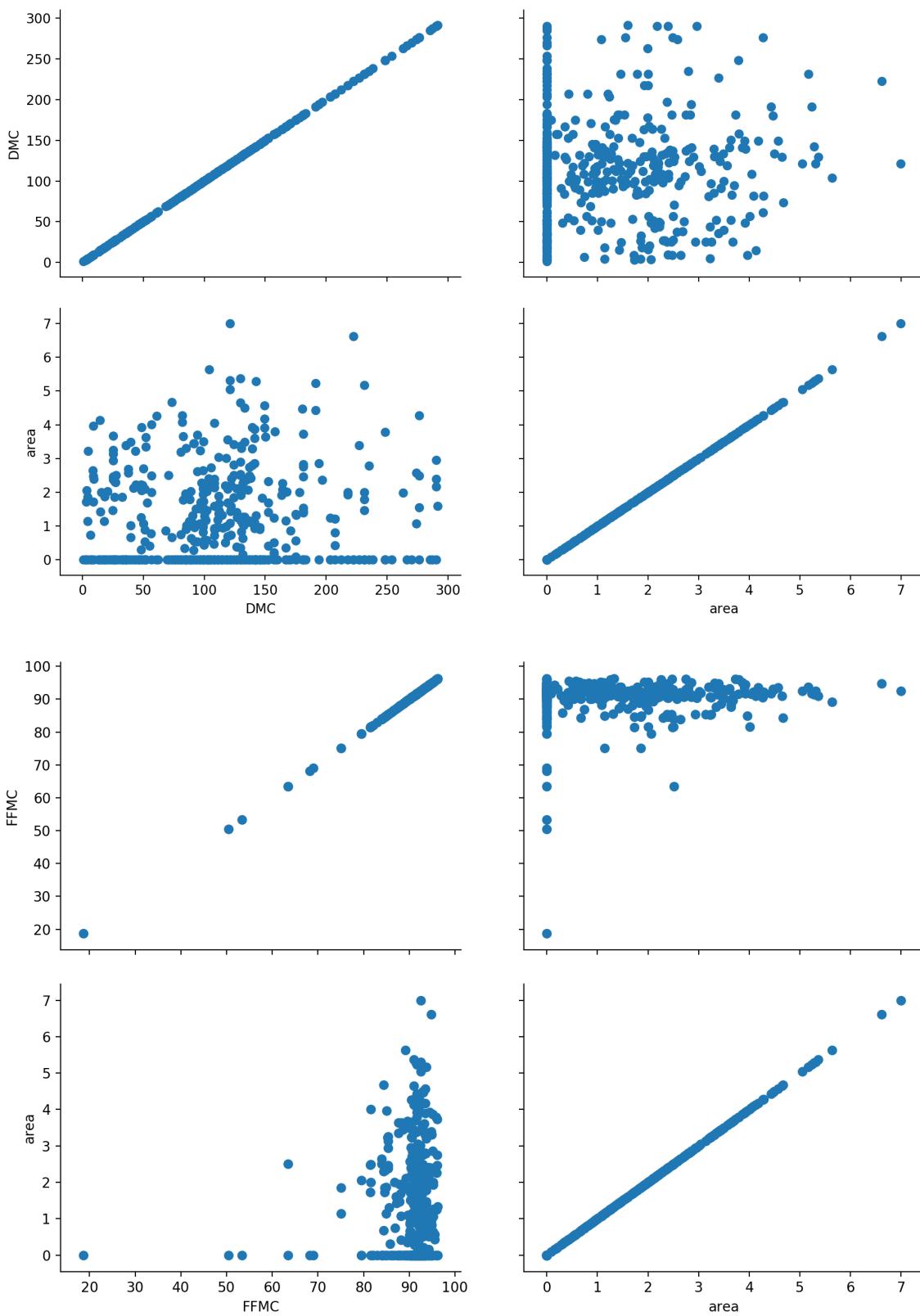
```
import pandas as pd
```

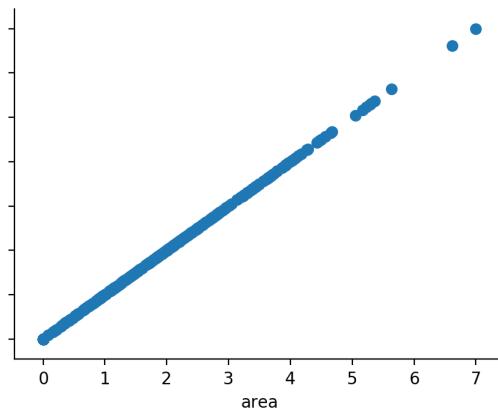
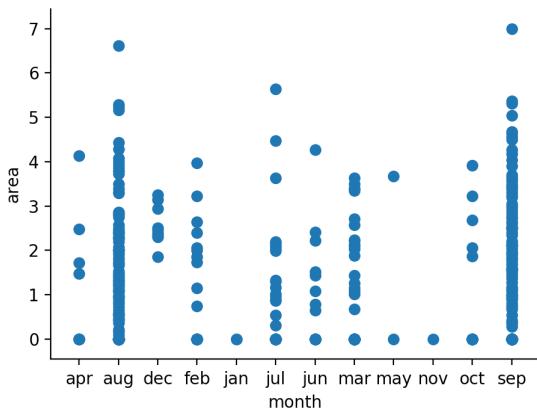
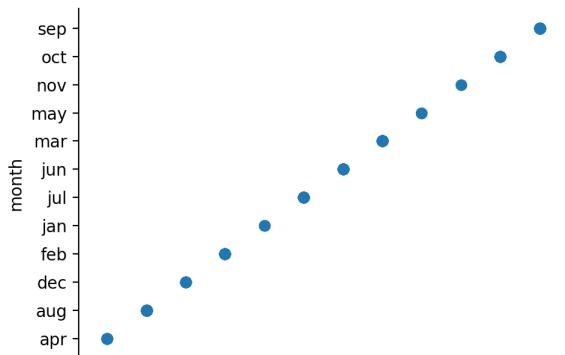
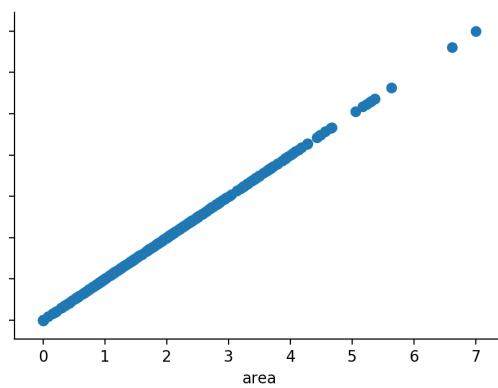
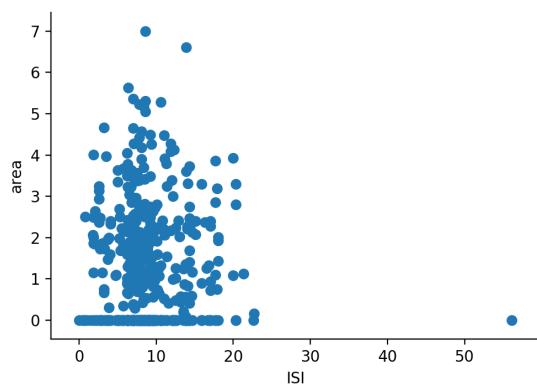
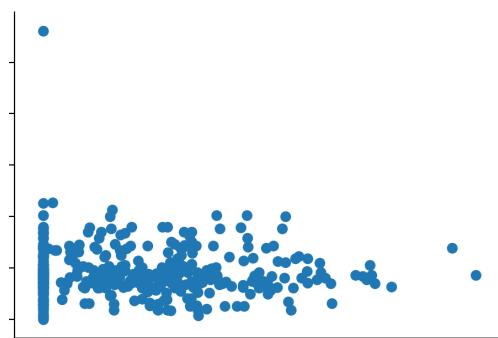
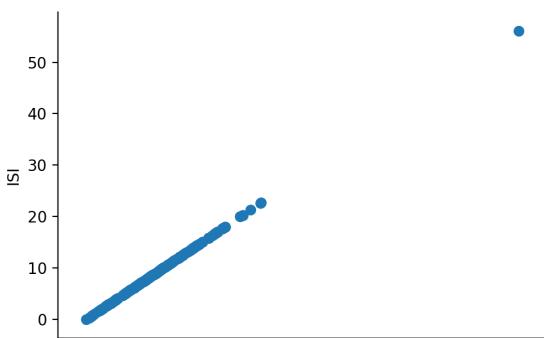
```
import math
import matplotlib.pyplot as plt
import seaborn as sns

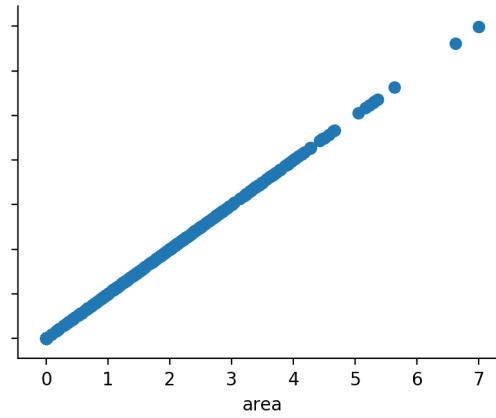
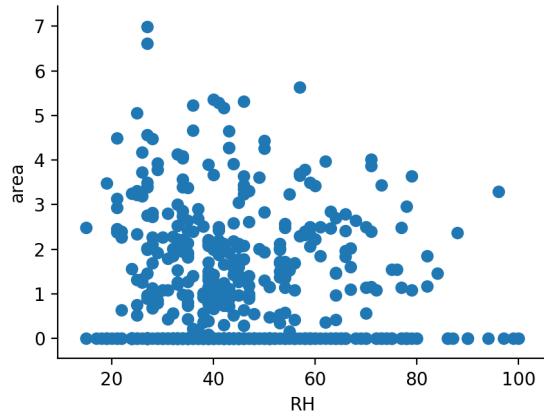
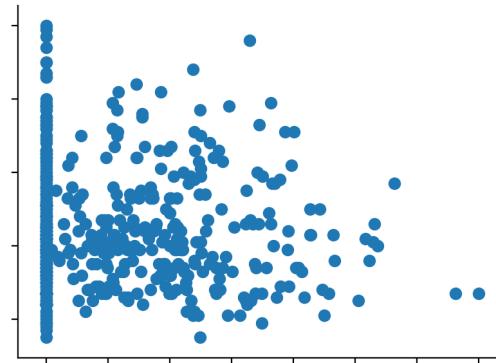
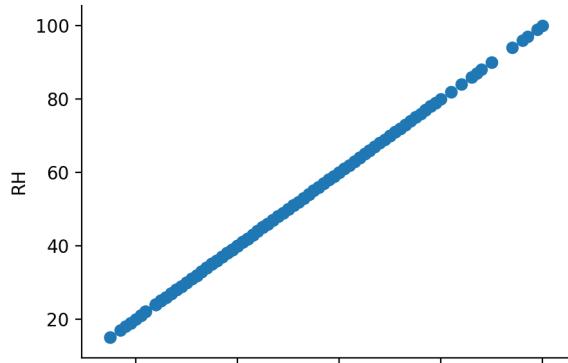
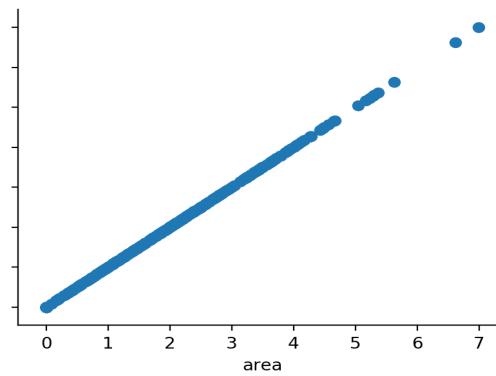
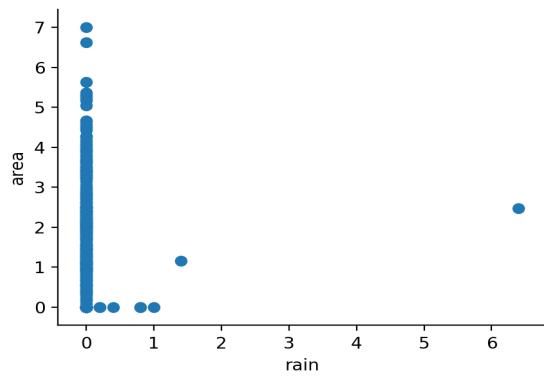
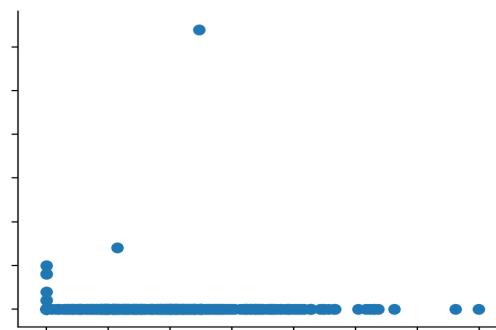
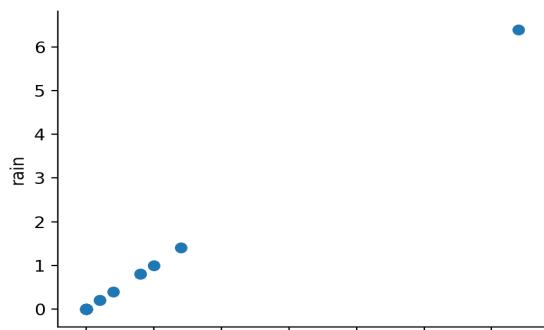
ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))
g = sns.PairGrid(ff_df, vars=["X", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["Y", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["month", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["day", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["FFMC", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["DMC", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["DC", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["ISI", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["temp", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["RH", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["wind", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
g = sns.PairGrid(ff_df, vars=["rain", "area"],size=3)
g = g.map(plt.scatter)
plt.show()
print(ff_df)
```

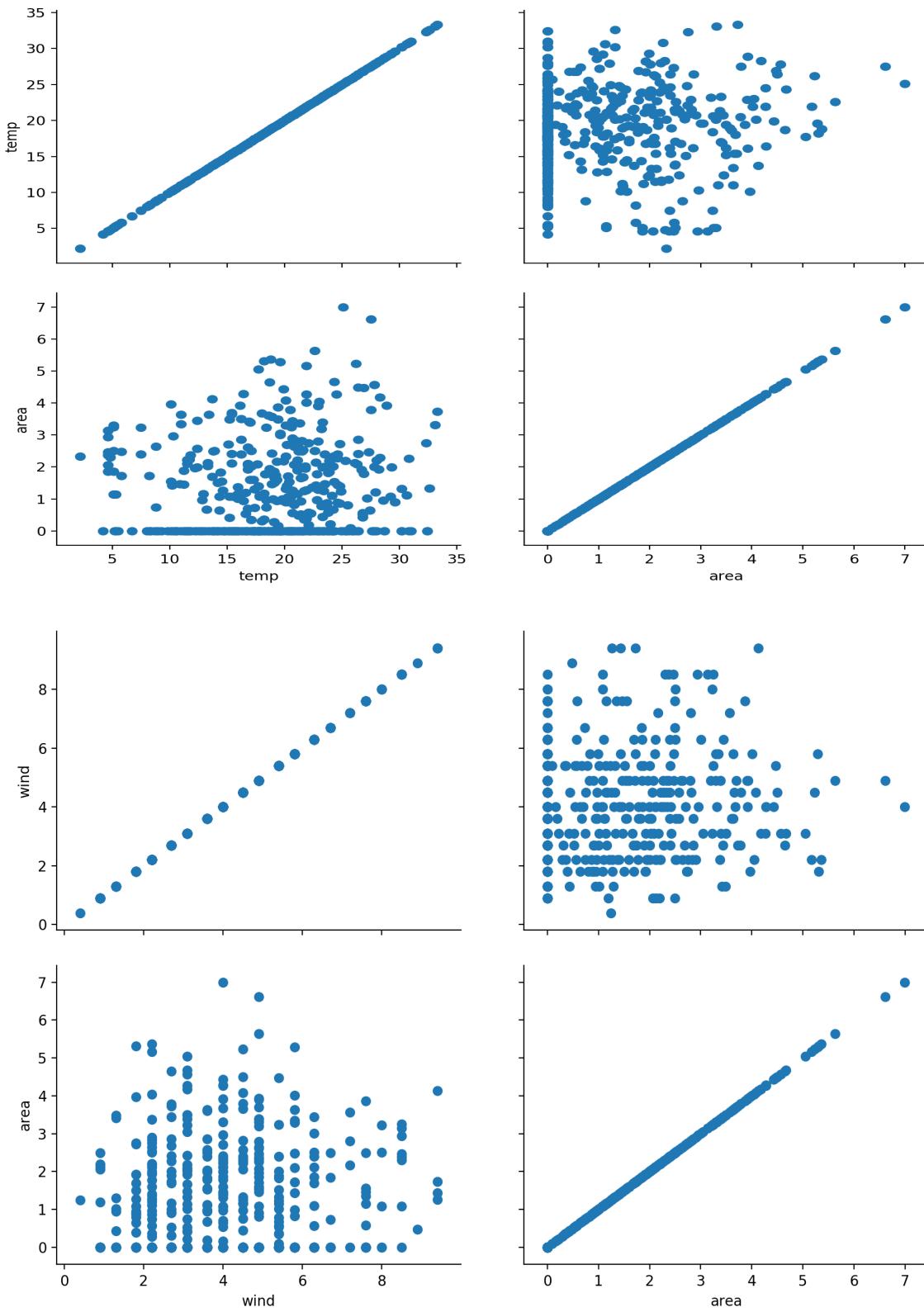
Output:

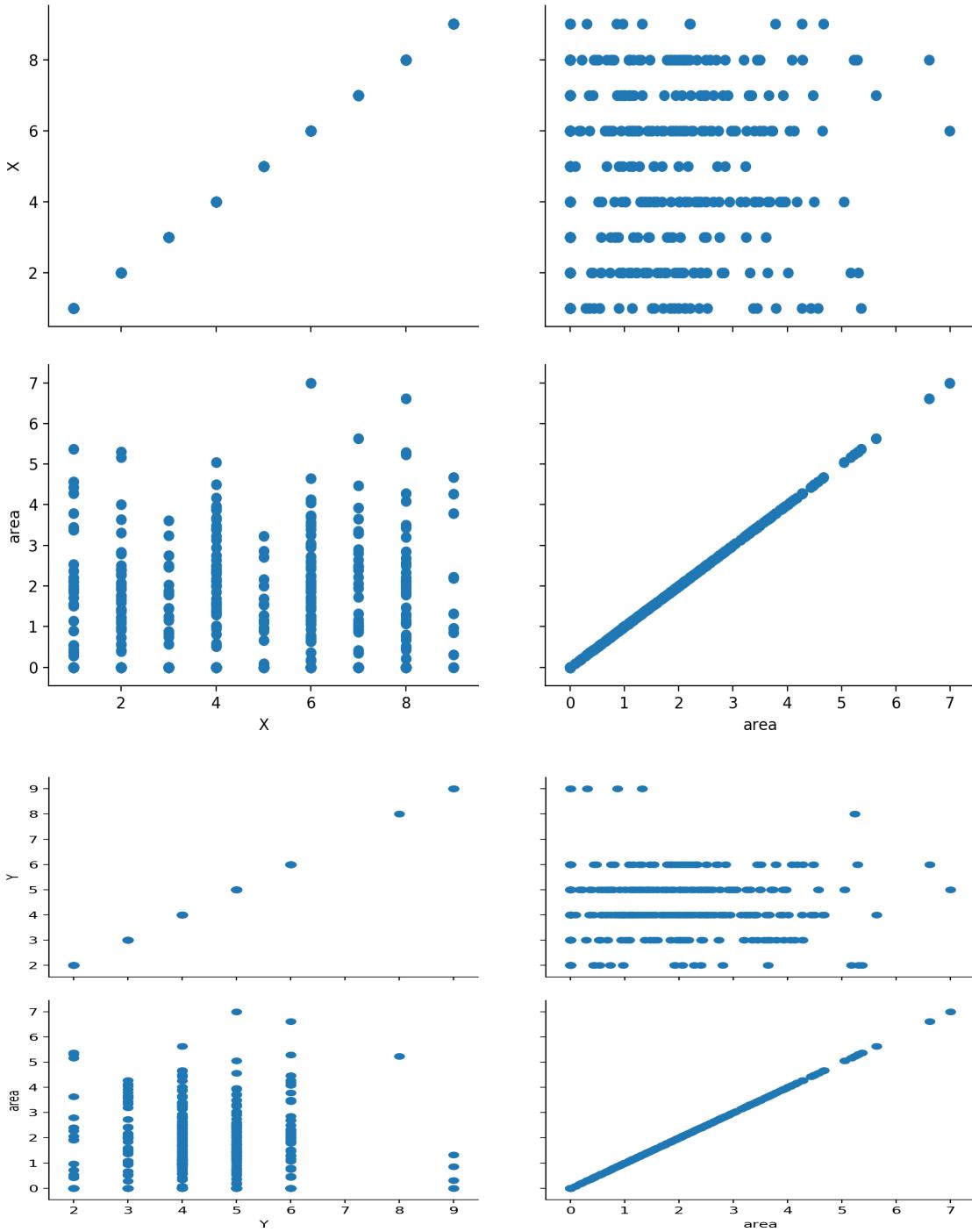












Inferences:

X and area :

Co-ordinates 3,5 and 9 look less prone to large forest fires. Other co-ordinates seem inconclusive.

Y and area:

Co-ordinates 7 and 8 have no data regarding forest fires and forest fires at co-ordinate 9 has not spread out to large areas.

Wind and area:

Very high and very low winds seem to have created only smaller forest fires , but wind levels between 2 and 6 have created a large forest fire.

Temp and area:

The relationship between temperature and area when there is a forest fire seems to be correlated to some extend. But since temperature alone is not a causal reason for forest fires the high temperature also records zero area which may have a negative effect on our regression model.

RH and area:

Low RH has always created large forest fires. Humidity seems to have a visible negative effect on forest fires.

Rain and area:

Rain has always been effective in suppressing forest fires to the extent that any rainy day can be for sure classified as a no forest fire or zero area.

Area and month:

August and September seem to have a lot of forest fires. Probably the climate is also hot(summer) and this has an effect on the areas covered. But again since the data points for no forest fire are also considered their overall effect on the mode might be inconclusive.

Area and ISI:

There is no visible correlation between ISI and area except for the fact that there are outliers in the data regarding ISI. Whether this is an important information or an outlier can be decided on statistical values in regressing these data.

Area and FFMC:

Again there is a heavy imbalance in data. This makes any possible correlation in this data impossible. But a high value of FFMC is always prone to large area.

Area and DMC:

No correlation visible in plot.

Area and DC:

There seems to be a lot of data for DC>600 and lesser data for DC<600. But there seems to be not much of correlation between area.

Area and day:

No correlation visible in plot.

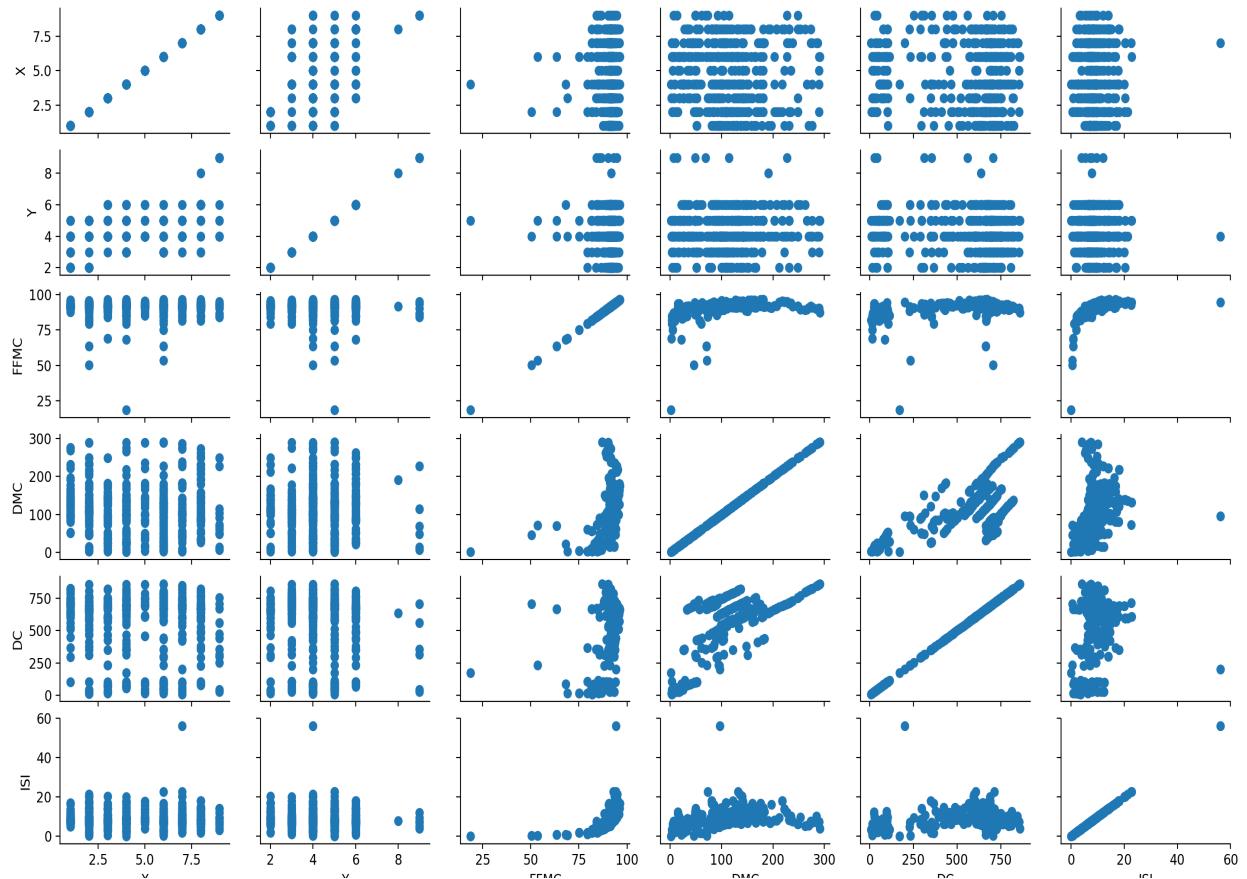
iv. Make at least 16 pairwise scatterplots of predictors of your choice and describe your findings. You are welcome to make all possible scatter plots.

Code:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))
g = sns.PairGrid(ff_df.iloc[:,0:8])
g = g.map(plt.scatter)
plt.show()
print(ff_df)
```

Output:



Inference:

The scatter plots show correlation between features which can be exploited for interaction. The pairs that have a good correlation are DMC:DC , FFMC:ISI. These hint us that we have to check for interaction between features which can help us in improving our model.

v. **What are the mean, the median, range, first and third quartiles, and interquartile ranges of each of the variables in the dataset? Summarize them in a table.**

Code:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))
describe = ff_df.describe()
#print(describe.loc["mean"])
table = pd.DataFrame()
table["Mean"] = describe.loc["mean"]
table["Median"] = ff_df.median()
#table["Range"] = describe.loc["max"] + " - " + describe.loc["min"]
table["Range"] = describe.loc["max"] - describe.loc["min"]
table["FirstQuartile"] = describe.loc["25%"]
table["ThirdQuartile"] = describe.loc["75%"]
table["Interquartile"] = describe.loc["75%"]-describe.loc["25%"]
print(final)
```

Output:

```
[Ashvants-MacBook-Pro:2B ashvant$ python3 HW1Q2B5.py
      mean    median     range fistQuartile thirdQuartile \
X      4.669246    4.00000    8.00000      3.0    7.000000
Y      4.299807    4.00000    7.00000      4.0    5.000000
FFMC   90.644681   91.60000   77.50000     90.2   92.900000
DMC   110.872340  108.30000  290.20000     68.6  142.400000
DC    547.940039  664.20000  852.70000    437.7 713.900000
ISI    9.021663    8.40000    56.10000      6.5   10.800000
temp   18.889168   19.30000   31.10000     15.5   22.800000
RH    44.288201   42.00000   85.00000     33.0   53.000000
wind   4.017602    4.00000    9.00000      2.7   4.900000
rain   0.021663    0.00000    6.40000      0.0   0.000000
area   1.111026    0.41871    6.99562      0.0   2.024193

      interquartile
X        4.000000
Y        1.000000
FFMC    2.700000
DMC    73.800000
DC    276.200000
ISI    4.300000
temp   7.300000
RH    20.000000
wind   2.200000
rain   0.000000
area   2.024193
Ashvants-MacBook-Pro:2B ashvant$
```

(c) For each predictor, fit a simple linear regression model to predict the response. Describe your results. In which of the models is there a statistically significant association between the predictor and the response? Create some plots to back up your assertions. Are there any outliers that you would like to remove from your data for each of these regression tasks?

Code:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
from statsmodels.formula.api import ols
from scipy import stats
import numpy as np
import seaborn as sns
ff_df = pd.read_csv("forestfires.csv")
#applying lambda transform
ff_df['area'] = ff_df['area'].apply(lambda x:math.log1p(x))

pvalues = []
rsquaredvalues = []
colnames=[]

##Linear regression for each variable
model = ols("area ~ X", ff_df).fit()
```

```

print(model.summary())
pvalues.append(model.pvalues["X"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ Y", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["Y"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ month", ff_df).fit()
print(model.summary())
rsquaredvalues.append(model.rsquared)
for key in list(model.pvalues)[1:]:
    pvalues.append(key)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ day", ff_df).fit()
print(model.summary())
rsquaredvalues.append(model.rsquared)
for key in list(model.pvalues)[1:]:
    pvalues.append(key)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ FFMC", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["FFMC"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ DMC", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["DMC"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ DC", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["DC"])

```

```

rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ ISI", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["ISI"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ temp", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["temp"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ RH", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["RH"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ wind", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["wind"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ rain", ff_df).fit()
print(model.summary())
pvalues.append(model.pvalues["rain"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

print(pvalues)
print(colnames)

##Graphs on p-values
y_pos = range(len(colnames))
a = plt.bar(y_pos, pvalues, align='center', alpha=0.5)

```

```

plt.xticks(y_pos, colnames, rotation='vertical')
plt.ylabel('p-values')
plt.title('predictors')
plt.show()

colnames2 = []
for column in ff_df.columns[0:12]:
    colnames2.append(column)

y_pos = range(len(colnames2))
##Graph on r-squared values
plt.bar(y_pos, rsquaredvalues, align='center', alpha=0.5)
plt.xticks(y_pos, colnames2, rotation='vertical')
plt.ylabel('r-squared values')
plt.title('predictors')
plt.show()

#Removing outliers for non -categorical variables
a_df = ff_df.loc[:, ff_df.columns != 'month']
new_df = a_df.loc[:, a_df.columns != 'day']
new_df = new_df.loc[:, new_df.columns != 'area']
new_df = new_df.loc[:, new_df.columns != 'rain']
low = .03
high = .97
quant_df = new_df.quantile([low, high])
new_df = new_df.apply(lambda x: x[(x >= quant_df.loc[low, x.name]) &
                                    (x <= quant_df.loc[high, x.name])], axis=0)
new_df = pd.concat([ff_df.loc[:, 'month'], new_df], axis=1)
new_df = pd.concat([ff_df.loc[:, 'day'], new_df], axis=1)
new_df = pd.concat([ff_df.loc[:, 'area'], new_df], axis=1)
new_df = pd.concat([ff_df.loc[:, 'rain'], new_df], axis=1)
new_df.dropna(inplace=True)
print(new_df)

#Boxplots to show to what extent outliers are removed
colnames = []
for column in ff_df.columns[0:2]:
    colnames.append(column)
for column in ff_df.columns[4:12]:
    colnames.append(column)
fig, ax = plt.subplots(nrows=2, ncols=10, squeeze=True)
j=0
for row in ax:
    i=0
    for col in row:

```

```

if j==0:
    col.boxplot(ff_df[colnames[i]].values)
if j==1:
    col.boxplot(new_df[colnames[i]].values)
i=i+1
j=j+1
plt.show()

```

#Converting rain to a categorical tackle data imbalance after removing outliers in other features - 0 dominates.

```

rainlist = np.array(new_df["rain"])
new_rainlist = []
for i in rainlist:
    if i==0:
        new_rainlist.append("No_rain")
    else:
        new_rainlist.append("Yes_rain")
new_df["rain"] = new_rainlist
print(new_df)

```

#Again running linear regression for data without outliers.

```

pvalues = []
rsquaredvalues = []
colnames=[]

##Linear regression for each variable
model = ols("area ~ X", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["X"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ Y", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["Y"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ month", new_df).fit()
print(model.summary())
rsquaredvalues.append(model.rsquared)

```

```

for key in list(model.pvalues)[1:]:
    pvalues.append(key)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ day", new_df).fit()
print(model.summary())
rsquaredvalues.append(model.rsquared)
for key in list(model.pvalues)[1:]:
    pvalues.append(key)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ FFMC", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["FFMC"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ DMC", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["DMC"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ DC", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["DC"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ ISI", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["ISI"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ temp", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["temp"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

```

```

colnames.append(key)

model = ols("area ~ RH", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["RH"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ wind", new_df).fit()
print(model.summary())
pvalues.append(model.pvalues["wind"])
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

model = ols("area ~ rain", new_df).fit()
print(model.summary())
for key in list(model.pvalues)[1:]:
    pvalues.append(key)
rsquaredvalues.append(model.rsquared)
for key in model.pvalues.keys()[1:]:
    colnames.append(key)

print(pvalues)
print(colnames)

##Graphs on p-values
y_pos = range(len(colnames))
a = plt.bar(y_pos, pvalues, align='center', alpha=0.5)
plt.xticks(y_pos, colnames, rotation='vertical')
plt.ylabel('p-values')
plt.title('predictors')
plt.show()

colnames2 = []
for column in ff_df.columns[0:12]:
    colnames2.append(column)

y_pos = range(len(colnames2))
##Graph on r-squared values
plt.bar(y_pos, rsquaredvalues, align='center', alpha=0.5)
plt.xticks(y_pos, colnames2, rotation='vertical')
plt.ylabel('r-squared values')

```

```
plt.title('predictors')
plt.show()
```

Output and Inferences:

X vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.004			
Model:	OLS	Adj. R-squared:	0.002			
Method:	Least Squares	F-statistic:	1.987			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.159			
Time:	08:10:50	Log-Likelihood:	-905.47			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1823.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.9361	0.138	6.759	0.000	0.664	1.208
X	0.0375	0.027	1.410	0.159	-0.015	0.090
Omnibus:	93.301	Durbin-Watson:	0.939			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	141.843			
Skew:	1.204	Prob(JB):	1.58e-31			
Kurtosis:	3.887	Cond. No.	12.1			

Normally p-value of 0.159 is not a strong a conclusion against the null hypothesis. But in this scenario relative p-values suggest that X is one of the factors that to an extent correlates with area.

Y vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.002			
Model:	OLS	Adj. R-squared:	-0.000			
Method:	Least Squares	F-statistic:	0.7780			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.378			
Time:	08:10:50	Log-Likelihood:	-906.08			
No. Observations:	517	AIC:	1816.			
Df Residuals:	515	BIC:	1825.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.9211	0.224	4.114	0.000	0.481	1.361
Y	0.0442	0.050	0.882	0.378	-0.054	0.143
Omnibus:	94.666	Durbin-Watson:	0.932			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	144.912			
Skew:	1.213	Prob(JB):	3.41e-32			
Kurtosis:	3.915	Cond. No.	17.0			

The p and t statistics suggest that null hypothesis cannot be ruled out for Y co-ordinates.

Month Vs Area

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.037			
Model:	OLS	Adj. R-squared:	0.016			
Method:	Least Squares	F-statistic:	1.765			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.0572			
Time:	08:10:50	Log-Likelihood:	-896.72			
No. Observations:	517	AIC:	1817.			
Df Residuals:	505	BIC:	1868.			
Df Model:	11					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.0892	0.462	2.356	0.019	0.181	1.998
month[T.aug]	-0.0440	0.474	-0.093	0.926	-0.974	0.886
month[T.dec]	1.4824	0.654	2.267	0.024	0.198	2.767
month[T.feb]	-0.0014	0.557	-0.002	0.998	-1.095	1.093
month[T.jan]	-1.0892	1.084	-1.004	0.316	-3.220	1.041
month[T.jul]	-0.0056	0.523	-0.011	0.991	-1.034	1.023
month[T.jun]	-0.2461	0.572	-0.430	0.667	-1.370	0.877
month[T.mar]	-0.3167	0.499	-0.634	0.526	-1.298	0.665
month[T.may]	0.7487	1.084	0.690	0.490	-1.382	2.879
month[T.nov]	-1.0892	1.462	-0.745	0.457	-3.962	1.784
month[T.oct]	-0.1722	0.585	-0.294	0.769	-1.321	0.977
month[T.sep]	0.1853	0.474	0.391	0.696	-0.747	1.117
Omnibus:	101.243	Durbin-Watson:	0.935			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	160.967			
Skew:	1.250	Prob(JB):	1.11e-35			
Kurtosis:	4.107	Cond. No.	32.8			

The one hot encoded month variables do not seem to have a correlation with area according to the suggested p-values.

Day vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.004			
Model:	OLS	Adj. R-squared:	-0.008			
Method:	Least Squares	F-statistic:	0.3568			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.906			
Time:	08:10:50	Log-Likelihood:	-905.39			
No. Observations:	517	AIC:	1825.			
Df Residuals:	510	BIC:	1855.			
Df Model:	6					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.9698	0.152	6.370	0.000	0.671	1.269
day[T.mon]	0.1202	0.223	0.539	0.590	-0.318	0.559
day[T.sat]	0.2566	0.216	1.188	0.235	-0.168	0.681
day[T.sun]	0.1543	0.210	0.736	0.462	-0.257	0.566
day[T.thu]	0.0565	0.236	0.240	0.811	-0.406	0.519
day[T.tue]	0.2609	0.232	1.123	0.262	-0.195	0.717
day[T.wed]	0.1439	0.244	0.589	0.556	-0.336	0.624

Omnibus:	91.365	Durbin-Watson:	0.926
Prob(Omnibus):	0.000	Jarque-Bera (JB):	137.596
Skew:	1.189	Prob(JB):	1.32e-30
Kurtosis:	3.854	Cond. No.	7.43

Even the p-values of days suggest nothing positive towards elimination of null hypothesis. The r-squared values also do not suggest any statistical significance.

FFMC Vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.002			
Model:	OLS	Adj. R-squared:	0.000			
Method:	Least Squares	F-statistic:	1.130			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.288			
Time:	08:10:50	Log-Likelihood:	-905.90			
No. Observations:	517	AIC:	1816.			
Df Residuals:	515	BIC:	1824.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.0364	1.013	0.036	0.971	-1.953	2.026
FFMC	0.0119	0.011	1.063	0.288	-0.010	0.034
Omnibus:	93.878	Durbin-Watson:	0.927			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	143.139			
Skew:	1.208	Prob(JB):	8.28e-32			
Kurtosis:	3.900	Cond. No.	1.50e+03			

Again, the p-values and t-statistic do not give an assertion to eliminate the null hypothesis. R squared value is also too low to create a statistical significance. Neither the coefficient value is high to create an impact.

DMC Vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.005			
Model:	OLS	Adj. R-squared:	0.003			
Method:	Least Squares	F-statistic:	2.333			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.127			
Time:	08:10:50	Log-Likelihood:	-905.30			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1823.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.9485	0.123	7.718	0.000	0.707	1.190
DMC	0.0015	0.001	1.527	0.127	-0.000	0.003

Omnibus:	92.181	Durbin-Watson:	0.923
Prob(Omnibus):	0.000	Jarque-Bera (JB):	139.319
Skew:	1.197	Prob(JB):	5.59e-31
Kurtosis:	3.855	Cond. No.	256.

Though the p-value suggests a high probability of the result to be close to the population mean , relatively , it gives an evidence of correlation when compared to other variables. Neither the coefficient value is high to create an impact.

DC vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.004			
Model:	OLS	Adj. R-squared:	0.002			
Method:	Least Squares	F-statistic:	2.278			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.132			
Time:	08:10:50	Log-Likelihood:	-905.33			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1823.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.9060	0.149	6.078	0.000	0.613	1.199
DC	0.0004	0.000	1.509	0.132	-0.000	0.001
Omnibus:	93.787	Durbin-Watson:	0.921			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	142.918			
Skew:	1.208	Prob(JB):	9.24e-32			
Kurtosis:	3.895	Cond. No.	1.46e+03			

This case is similar to DMC

ISI vs Area:

OLS Regression Results						
Dep. Variable:	area	R-squared:	0.000			
Model:	OLS	Adj. R-squared:	-0.002			
Method:	Least Squares	F-statistic:	0.05514			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.814			
Time:	08:10:50	Log-Likelihood:	-906.44			
No. Observations:	517	AIC:	1817.			
Df Residuals:	515	BIC:	1825.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	1.1397	0.137	8.344	0.000	0.871	1.408
ISI	-0.0032	0.014	-0.235	0.814	-0.030	0.023
Omnibus:	94.904	Durbin-Watson:	0.925			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	145.494			
Skew:	1.214	Prob(JB):	2.55e-32			
Kurtosis:	3.927	Cond. No.	22.6			

The p-value and coefficient suggests that null hypothesis is very much possible for ISI.

Temp vs Area:

Warnings:						
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.						
OLS Regression Results						
=====						
Dep. Variable:	area	R-squared:	0.003			
Model:	OLS	Adj. R-squared:	0.001			
Method:	Least Squares	F-statistic:	1.478			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.225			
Time:	08:10:50	Log-Likelihood:	-905.73			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1824.			
Df Model:	1					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.8677	0.209	4.144	0.000	0.456	1.279
temp	0.0129	0.011	1.216	0.225	-0.008	0.034
=====						
Omnibus:	91.757	Durbin-Watson:	0.923			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	138.327			
Skew:	1.197	Prob(JB):	9.18e-31			
Kurtosis:	3.831	Cond. No.	67.5			
=====						

The co-efficient and p-values do not suggest any vital statistical relationship.

RH Vs Area

OLS Regression Results						
=====						
Dep. Variable:	area	R-squared:	0.003			
Model:	OLS	Adj. R-squared:	0.001			
Method:	Least Squares	F-statistic:	1.487			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.223			
Time:	08:10:50	Log-Likelihood:	-905.72			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1824.			
Df Model:	1					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.3147	0.178	7.387	0.000	0.965	1.664
RH	-0.0046	0.004	-1.220	0.223	-0.012	0.003
=====						
Omnibus:	92.444	Durbin-Watson:	0.927			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	139.921			
Skew:	1.199	Prob(JB):	4.14e-31			
Kurtosis:	3.866	Cond. No.	137.			
=====						

No statistical significance .. very insignificant co-efficient value and high p-value. Null hypothesis cannot be eliminated.

Wind vs Area

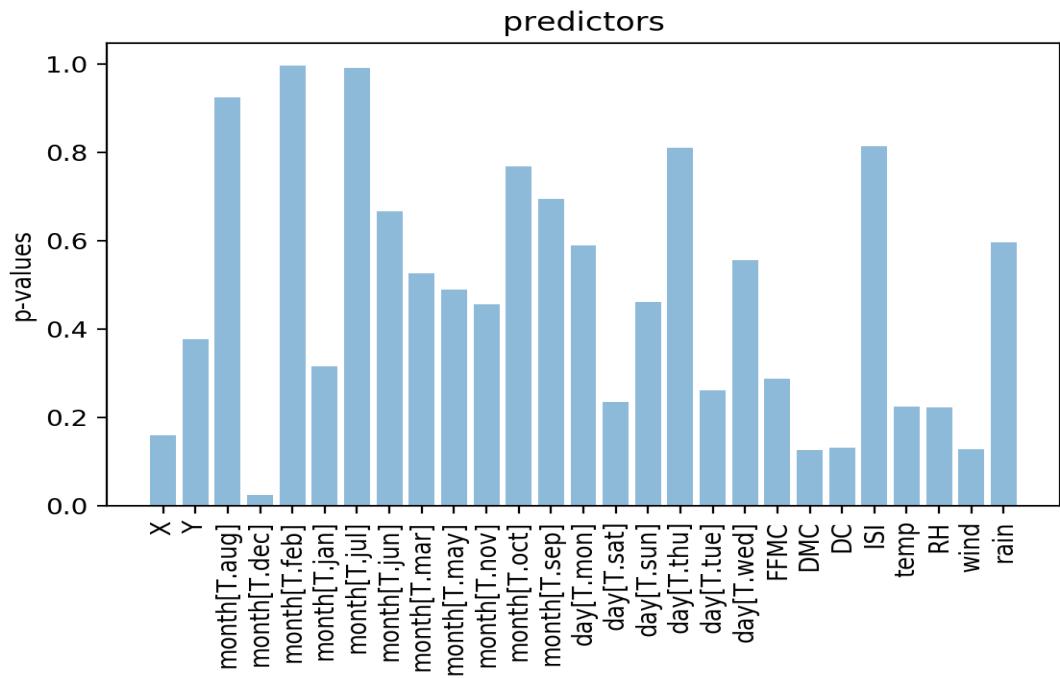
OLS Regression Results						
Dep. Variable:	area	R-squared:	0.004			
Model:	OLS	Adj. R-squared:	0.003			
Method:	Least Squares	F-statistic:	2.320			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.128			
Time:	08:10:50	Log-Likelihood:	-905.31			
No. Observations:	517	AIC:	1815.			
Df Residuals:	515	BIC:	1823.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	0.9010	0.151	5.969	0.000	0.604	1.198
wind	0.0523	0.034	1.523	0.128	-0.015	0.120
Omnibus:	95.685	Durbin-Watson:	0.926			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	147.420			
Skew:	1.216	Prob(JB):	9.73e-33			
Kurtosis:	3.962	Cond. No.	11.3			

Relatively better co-efficient and p-value. Can be further investigated for interaction and polynomial regression.

Rain Vs Area:

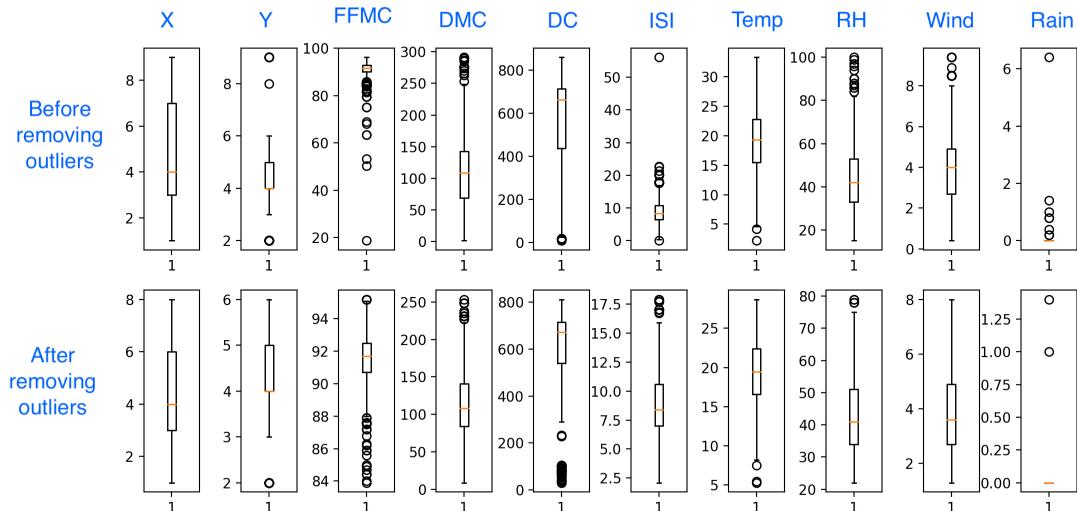
OLS Regression Results						
Dep. Variable:	area	R-squared:	0.001			
Model:	OLS	Adj. R-squared:	-0.001			
Method:	Least Squares	F-statistic:	0.2800			
Date:	Wed, 31 Jan 2018	Prob (F-statistic):	0.597			
Time:	08:10:50	Log-Likelihood:	-906.33			
No. Observations:	517	AIC:	1817.			
Df Residuals:	515	BIC:	1825.			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	1.1086	0.062	17.965	0.000	0.987	1.230
rain	0.1101	0.208	0.529	0.597	-0.299	0.519
Omnibus:	95.434	Durbin-Watson:	0.927			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	146.699			
Skew:	1.218	Prob(JB):	1.40e-32			
Kurtosis:	3.937	Cond. No.	3.38			

Though co-efficients are relatively high . High p-value suggests a null hypothesis may be present. Hence, no statistical conclusion can be made.



The above plot shows the p-values for various predictors and it is very much conclusive that none of the variables have a strong statistical significance, that is, null hypothesis cannot be eliminated.

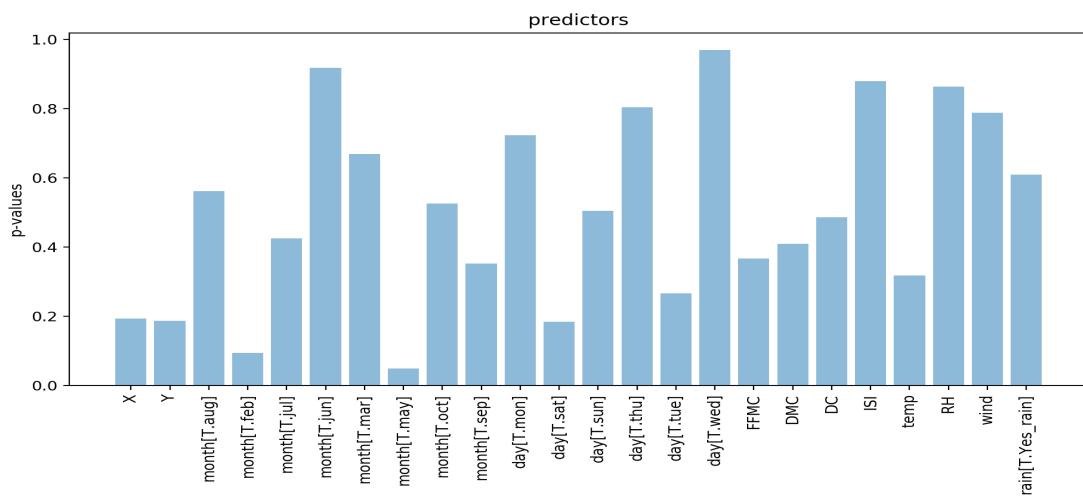
To estimate the outliers, we make a box plot before and after the outliers are removed for each variable:



The above plots show the removal of outliers. There are few outliers still present , but it is important to note that the range of the box plots differ in the two cases meaning these are newly formed outliers.

But removing outliers do not necessarily have to have positive effect on our model. Sometimes outliers carry important information.

Hence we run a linear regression again on the data after removing outliers and plot the p-values to see if there are any positive effects.



It is evident that removing outliers did not have a very positive effect on the p-values of each of the predictors.

(d) Fit a multiple regression model to predict the response using all of the predictors. Describe your results. For which predictors can we reject the null hypothesis $H_0: \beta_j = 0$?

Code:

```

import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

model = ols("area ~ X+Y+month+day+FFMC+DMC+DC+ISI+temp+RH+wind+rain",
ff_df).fit()

```

```

print(model.summary())

pvalues = []
keys = []
intercept=[]

for key in model.pvalues.keys()[1:]:
    intercept.append(model.params[key])
    keys.append(key)
    pvalues.append(model.pvalues[key])

y_pos = range(len(keys))
a = plt.bar(y_pos, intercept, align='center', alpha=0.5)
plt.xticks(y_pos, keys, rotation='vertical')
plt.ylabel('intercept values')
plt.title('predictors')
plt.show()

y_pos = range(len(keys))
a = plt.bar(y_pos, pvalues, align='center', alpha=0.5)
plt.xticks(y_pos, keys, rotation='vertical')
plt.ylabel('p-values')
plt.title('predictors')
plt.show()

```

Output:

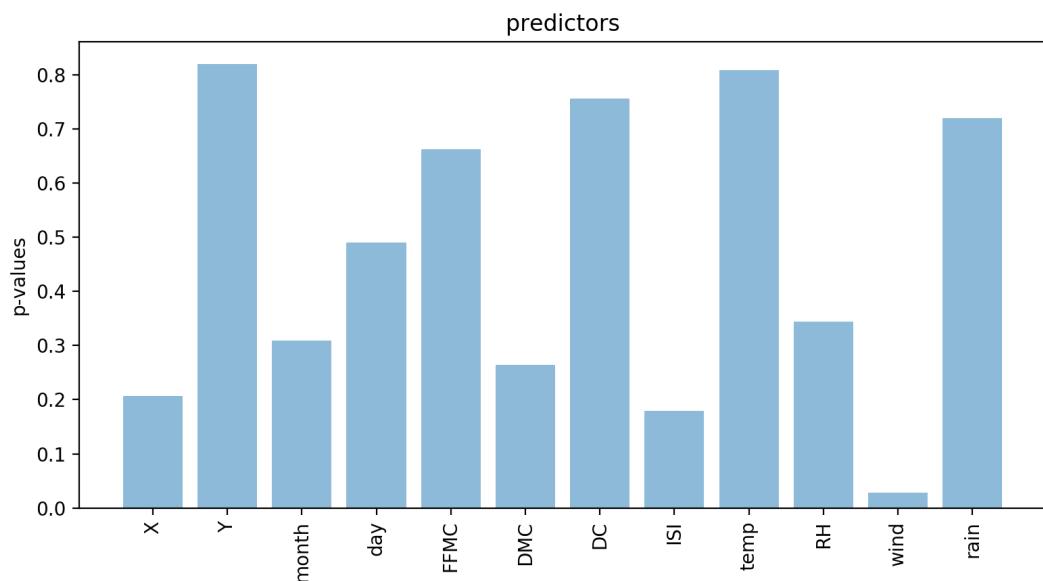
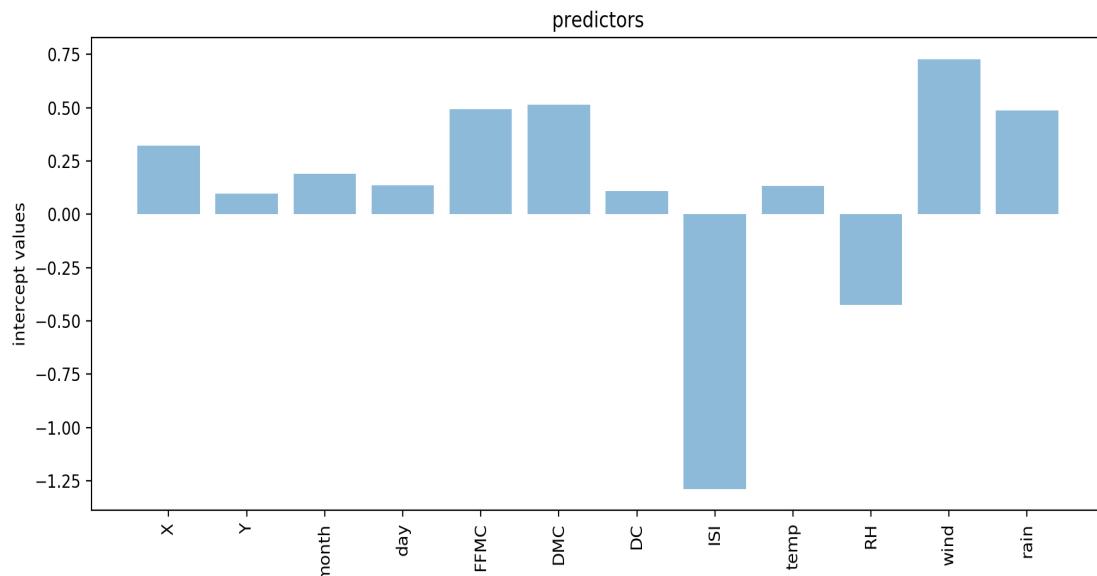
OLS Regression Results

Dep. Variable:	area	R-squared:	0.028			
Model:	OLS	Adj. R-squared:	0.005			
Method:	Least Squares	F-statistic:	1.207			
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.275			
Time:	02:18:38	Log-Likelihood:	-899.15			
No. Observations:	517	AIC:	1824.			
Df Residuals:	504	BIC:	1880.			
Df Model:	12					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.0388	1.071	0.036	0.971	-2.065	2.142
X	0.3213	0.254	1.263	0.207	-0.178	0.821
Y	0.0957	0.421	0.227	0.820	-0.731	0.922
month	0.1905	0.187	1.018	0.309	-0.177	0.558
day	0.1361	0.197	0.691	0.490	-0.251	0.523
FFMC	0.4920	1.128	0.436	0.663	-1.724	2.708
DMC	0.5131	0.459	1.119	0.264	-0.388	1.414
DC	0.1089	0.351	0.310	0.757	-0.581	0.799
ISI	-1.2887	0.959	-1.344	0.179	-3.172	0.595
temp	0.1323	0.547	0.242	0.809	-0.943	1.207
RH	-0.4245	0.448	-0.948	0.344	-1.305	0.456
wind	0.7279	0.332	2.194	0.029	0.076	1.380
rain	0.4870	1.361	0.358	0.721	-2.188	3.162

Omnibus:	84.922	Durbin-Watson:	0.941
Prob(Omnibus):	0.000	Jarque-Bera (JB):	123.977
Skew:	1.142	Prob(JB):	1.20e-27
Kurtosis:	3.731	Cond. No.	52.1

The summary of a multiple linear regression on all variables.

Below are the plots of the co-efficients and p-values of the variable.



A multiple Linear Regression requires that all features are considered. Hence it is important that we normalize the variables to avoid possible bias towards a value that has higher magnitude. I have also label encoded the categorical variables for better interpretation.

The null hypothesis can be eliminated for the feature wind. Other p-values are higher than 0.1. Hence null hypothesis cannot be eliminated for other variables. Wind also has a high co-efficient supporting the claim that wind has a considerable effect on the area of forest fires.

Note: For this and the following questions the variables are normalized and categorical variables are label encoded for better interpretability.

(e) How do your results from 2c compare to your results from 2d? Create a plot displaying the univariate regression coefficients from 2c on the x-axis, and the multiple regression coefficients from 2d on the y-axis. That is, each predictor is displayed as a single point in the plot. Its coefficient in a simple linear regression model is shown on the x-axis, and its coefficient estimate in the multiple linear regression model is shown on the y-axis.

Code:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn import preprocessing

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"]=le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"]=le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]

model = ols("area ~ X+Y+month+day+FFMC+DMC+DC+ISI+temp+RH+wind+rain",
normalized_df).fit()
#print(list(model.params)[1:])

multiple = dict()
```

```

for key in model.params.keys()[1:]:
    multiple[key] = model.params[key]
#print(multiple)

single=dict()
model = ols("area ~ X", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ Y", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ month", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ day", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ FFMC", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ DMC", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ DC", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ ISI", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ temp", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ RH", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]

model = ols("area ~ wind", normalized_df).fit()

```

```

for key in model.params.keys()[1:]:
    single[key] = model.params[key]

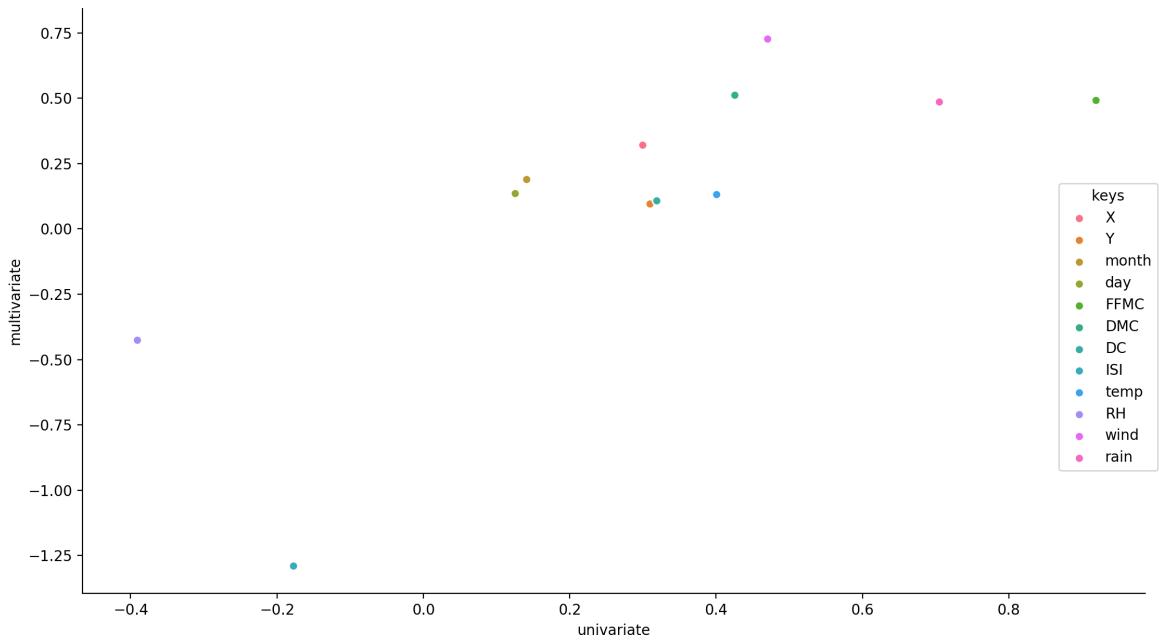
model = ols("area ~ rain", normalized_df).fit()
for key in model.params.keys()[1:]:
    single[key] = model.params[key]
#print(single)

singleList = []
multipleList = []
keyList = []
for key in single.keys():
    singleList.append(single[key])
    multipleList.append(multiple[key])
    keyList.append(key)

df = pd.DataFrame()
df["univariate"] = singleList
df["multivariate"] = multipleList
df["keys"] = keyList
print(df)
sns.pairplot(x_vars=["univariate"], y_vars=["multivariate"], data=df, hue="keys", size=5)
plt.show()

```

Output:



Comparing the Rsquared values of individual regression and multiple regression, there has been a considerable increase as the features together have improved the model. Hence, we continue with using multiple regression and improve the model with interaction and polynomial regression.

The co-efficient values have visible broadened outwards from 0 and hence created a greater impact in multiple regression. But still most features have not shown any correlation and hence it is important to try out polynomial regression of variables.

(f) Is there evidence of nonlinear association between any of the predictors and the response? To answer this question, for each predictor X, fit a model of the form $Y_1 = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 +$

CODE:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn import preprocessing

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"] = le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"] = le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]

pvalues=[]
keys=[]
intercept = []
for column in list(ff_df.columns[:-1]):

    print("*****"+column.upper()+"*****")
    print("*****")
```

```

model = ols("area ~ "+column+" + I("+column+"** 2.0) + I("+column+"**3.0)",
normalized_df).fit()
print(model.summary())
for key in model.pvalues.keys()[1:]:
    intercept.append(model.params[key])
    pvalues.append(model.pvalues[key])
    keys.append(column+"-"+key)

##Graphs on p-values
y_pos = range(len(keys))
a = plt.bar(y_pos, pvalues, align='center', alpha=0.5)
plt.xticks(y_pos, keys, rotation='vertical')
plt.ylabel('p-values')
plt.title('predictors')
plt.show()

##Graphs on intercept values
y_pos = range(len(keys))
a = plt.bar(y_pos, intercept, align='center', alpha=0.5)
plt.xticks(y_pos, keys, rotation='vertical')
plt.ylabel('intercept')
plt.title('predictors')
plt.show()

```

Output:

```

*****
OLS Regression Results
*****
Dep. Variable: area R-squared: 0.007
Model: OLS Adj. R-squared: 0.002
Method: Least Squares F-statistic: 1.281
Date: Sat, 03 Feb 2018 Prob (F-statistic): 0.280
Time: 02:41:57 Log-Likelihood: -904.54
No. Observations: 517 AIC: 1817.
Df Residuals: 513 BIC: 1834.
Df Model: 3
Covariance Type: nonrobust
*****
            coef  std err      t      P>|t|      [0.025]      [0.975]
Intercept  1.1370   0.183    6.226    0.000     0.778     1.496
X         -1.0317   1.686   -0.612    0.541    -4.344     2.280
I(X ** 2.0)  1.9757   4.211    0.469    0.639    -6.297    10.248
I(X ** 3.0) -0.6284   2.932   -0.214    0.830    -6.388     5.131
*****
Omnibus: 91.338 Durbin-Watson: 0.948
Prob(Omnibus): 0.000 Jarque-Bera (JB): 137.489
Skew: 1.191 Prob(JB): 1.40e-30
Kurtosis: 3.842 Cond. No. 103.
*****
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****
OLS Regression Results
*****
```

Dep. Variable:	area	R-squared:	0.004
Model:	OLS	Adj. R-squared:	-0.002
Method:	Least Squares	F-statistic:	0.6123
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.607
Time:	02:41:57	Log-Likelihood:	-905.54
No. Observations:	517	AIC:	1819.
Df Residuals:	513	BIC:	1836.
Df Model:	3		
Covariance Type:	nonrobust		
coef	std err	t	P> t
Intercept	0.9271	0.199	4.664
Y	0.6619	1.779	0.372
I(Y ** 2.0)	0.2585	4.789	0.054
I(Y ** 3.0)	-1.0194	3.497	-0.292
Omnibus:	96.141	Durbin-Watson:	0.938
Prob(Omnibus):	0.000	Jarque-Bera (JB):	148.391
Skew:	1.221	Prob(JB):	5.99e-33
Kurtosis:	3.961	Cond. No.	106.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****MONTH*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.013
Model:	OLS	Adj. R-squared:	0.008
Method:	Least Squares	F-statistic:	2.334
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.0732
Time:	02:41:57	Log-Likelihood:	-902.96
No. Observations:	517	AIC:	1814.
Df Residuals:	513	BIC:	1831.
Df Model:	3		
Covariance Type:	nonrobust		
coef	std err	t	P> t
Intercept	0.7981	0.262	3.045
month	4.3443	3.129	1.388
I(month ** 2.0)	-12.1021	7.317	-1.654
I(month ** 3.0)	8.2340	4.494	1.832
Omnibus:	93.415	Durbin-Watson:	0.926
Prob(Omnibus):	0.000	Jarque-Bera (JB):	142.233
Skew:	1.201	Prob(JB):	1.30e-31
Kurtosis:	3.912	Cond. No.	203.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****DAY*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.002
Model:	OLS	Adj. R-squared:	-0.004
Method:	Least Squares	F-statistic:	0.3696
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.775
Time:	02:41:57	Log-Likelihood:	-905.91
No. Observations:	517	AIC:	1820.
Df Residuals:	513	BIC:	1837.
Df Model:	3		

Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.9710	0.147	6.603	0.000	0.682	1.260
day	1.2083	1.444	0.837	0.403	-1.630	4.046
I(day ** 2.0)	-2.3576	3.685	-0.640	0.523	-9.598	4.882
I(day ** 3.0)	1.3364	2.479	0.539	0.590	-3.534	6.207

Omnibus:	92.733	Durbin-Watson:	0.930
Prob(Omnibus):	0.000	Jarque-Bera (JB):	140.572
Skew:	1.200	Prob(JB):	2.99e-31
Kurtosis:	3.874	Cond. No.	90.5

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****FFMC*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.003
Model:	OLS	Adj. R-squared:	-0.003
Method:	Least Squares	F-statistic:	0.4899
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.689
Time:	02:41:57	Log-Likelihood:	-905.73
No. Observations:	517	AIC:	1819.
Df Residuals:	513	BIC:	1836.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.1225	1.394	-0.088	0.930	-2.862	2.617
FFMC	0.3245	8.822	0.037	0.971	-17.006	17.656
I(FFMC ** 2.0)	3.5876	17.104	0.210	0.834	-30.015	37.191
I(FFMC ** 3.0)	-2.6827	9.420	-0.285	0.776	-21.190	15.825

Omnibus:	94.270	Durbin-Watson:	0.925
Prob(Omnibus):	0.000	Jarque-Bera (JB):	144.077
Skew:	1.209	Prob(JB):	5.18e-32
Kurtosis:	3.917	Cond. No.	627.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****DMC*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.005
Model:	OLS	Adj. R-squared:	-0.001
Method:	Least Squares	F-statistic:	0.8103
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.489
Time:	02:41:57	Log-Likelihood:	-905.25
No. Observations:	517	AIC:	1818.
Df Residuals:	513	BIC:	1835.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.9465	0.232	4.085	0.000	0.491	1.402
DMC	0.2450	2.129	0.115	0.908	-3.937	4.427
I(DMC ** 2.0)	0.8333	5.446	0.153	0.878	-9.866	11.532
I(DMC ** 3.0)	-0.7630	3.824	-0.200	0.842	-8.275	6.749

Omnibus:	91.691	Durbin-Watson:	0.925
Prob(Omnibus):	0.000	Jarque-Bera (JB):	138.224
Skew:	1.195	Prob(JB):	9.66e-31
Kurtosis:	3.841	Cond. No.	124.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

*****DC*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.007
Model:	OLS	Adj. R-squared:	0.001
Method:	Least Squares	F-statistic:	1.126
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.338
Time:	02:41:57	Log-Likelihood:	-904.77
No. Observations:	517	AIC:	1818.
Df Residuals:	513	BIC:	1835.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.7122	0.243	2.934	0.003	0.235	1.189
DC	3.1777	2.733	1.163	0.245	-2.191	8.546
I(DC ** 2.0)	-6.5128	6.376	-1.021	0.308	-19.039	6.014
I(DC ** 3.0)	3.9958	4.114	0.971	0.332	-4.087	12.078

Omnibus:	95.146	Durbin-Watson:	0.920
Prob(Omnibus):	0.000	Jarque-Bera (JB):	146.063
Skew:	1.215	Prob(JB):	1.92e-32
Kurtosis:	3.934	Cond. No.	179.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

*****ISI*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.001
Model:	OLS	Adj. R-squared:	-0.005
Method:	Least Squares	F-statistic:	0.2203
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.882
Time:	02:41:57	Log-Likelihood:	-906.14
No. Observations:	517	AIC:	1820.
Df Residuals:	513	BIC:	1837.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.0489	0.294	3.566	0.000	0.471	1.627
ISI	0.7250	3.885	0.187	0.852	-6.907	8.357
I(ISI ** 2.0)	-1.6549	13.803	-0.120	0.905	-28.772	25.462
I(ISI ** 3.0)	-0.0920	10.576	-0.009	0.993	-20.870	20.686

Omnibus:	94.468	Durbin-Watson:	0.929
Prob(Omnibus):	0.000	Jarque-Bera (JB):	144.490
Skew:	1.211	Prob(JB):	4.21e-32
Kurtosis:	3.916	Cond. No.	292.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

*****TEMP*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.024
Model:	OLS	Adj. R-squared:	0.019
Method:	Least Squares	F-statistic:	4.278
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.00536
Time:	02:41:57	Log-Likelihood:	-900.08
No. Observations:	517	AIC:	1808.
Df Residuals:	513	BIC:	1825.
Df Model:	3		
Covariance Type:	nonrobust		
coef	std err	t	P> t
Intercept	2.2780	0.487	4.677
temp	-8.4590	3.455	-2.448
I(temp ** 2.0)	15.4644	7.428	2.082
I(temp ** 3.0)	-7.7685	4.814	-1.614
Omnibus:	94.948	Durbin-Watson:	0.955
Prob(Omnibus):	0.000	Jarque-Bera (JB):	145.652
Skew:	1.213	Prob(JB):	2.36e-32
Kurtosis:	3.936	Cond. No.	188.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****RH*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.005
Model:	OLS	Adj. R-squared:	-0.001
Method:	Least Squares	F-statistic:	0.7968
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.496
Time:	02:41:57	Log-Likelihood:	-905.27
No. Observations:	517	AIC:	1819.
Df Residuals:	513	BIC:	1836.
Df Model:	3		
Covariance Type:	nonrobust		
coef	std err	t	P> t
Intercept	1.3699	0.335	4.090
RH	-1.9878	2.812	-0.707
I(RH ** 2.0)	4.9270	6.831	0.721
I(RH ** 3.0)	-4.0014	4.846	-0.826
Omnibus:	93.062	Durbin-Watson:	0.933
Prob(Omnibus):	0.000	Jarque-Bera (JB):	141.360
Skew:	1.201	Prob(JB):	2.01e-31
Kurtosis:	3.891	Cond. No.	154.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
*****WIND*****

OLS Regression Results

Dep. Variable:	area	R-squared:	0.012
Model:	OLS	Adj. R-squared:	0.007
Method:	Least Squares	F-statistic:	2.157
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.0921
Time:	02:41:57	Log-Likelihood:	-903.23
No. Observations:	517	AIC:	1814.
Df Residuals:	513	BIC:	1831.

Df Model:
Covariance Type:

3
nonrobust

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.3865	0.386	1.001	0.317	-0.372	1.145
wind	5.7169	3.023	1.891	0.059	-0.222	11.656
I(wind ** 2.0)	-13.2356	6.908	-1.916	0.056	-26.807	0.336
I(wind ** 3.0)	9.2944	4.635	2.005	0.045	0.188	18.401

Omnibus:	96.599	Durbin-Watson:	0.933
Prob(Omnibus):	0.000	Jarque-Bera (JB):	149.584
Skew:	1.222	Prob(JB):	3.30e-33
Kurtosis:	3.986	Cond. No.	159.

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

*****RAIN*****

OLS Regression Results

Dep. Variable:

area R-squared: 0.009

Model:

OLS Adj. R-squared: 0.003

Method:

Least Squares F-statistic: 1.527

Date:

Sat, 03 Feb 2018 Prob (F-statistic): 0.206

Time:

02:41:57 Log-Likelihood: -904.17

No. Observations:

517 AIC: 1816.

Df Residuals:

513 BIC: 1833.

Df Model:

3

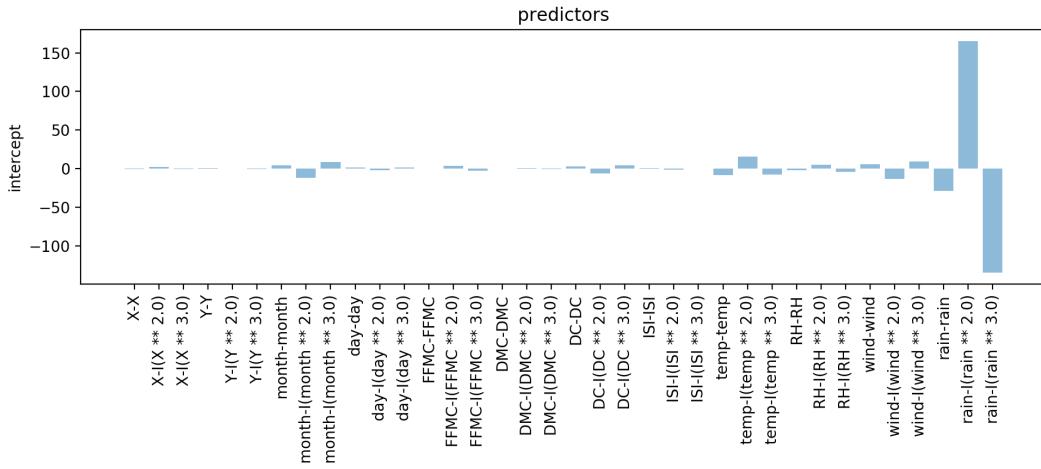
Covariance Type:

nonrobust

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.1205	0.062	18.116	0.000	0.999	1.242
rain	-29.0896	17.230	-1.688	0.092	-62.940	4.761
I(rain ** 2.0)	164.9341	117.278	1.406	0.160	-65.470	395.339
I(rain ** 3.0)	-134.4958	100.729	-1.335	0.182	-332.388	63.397

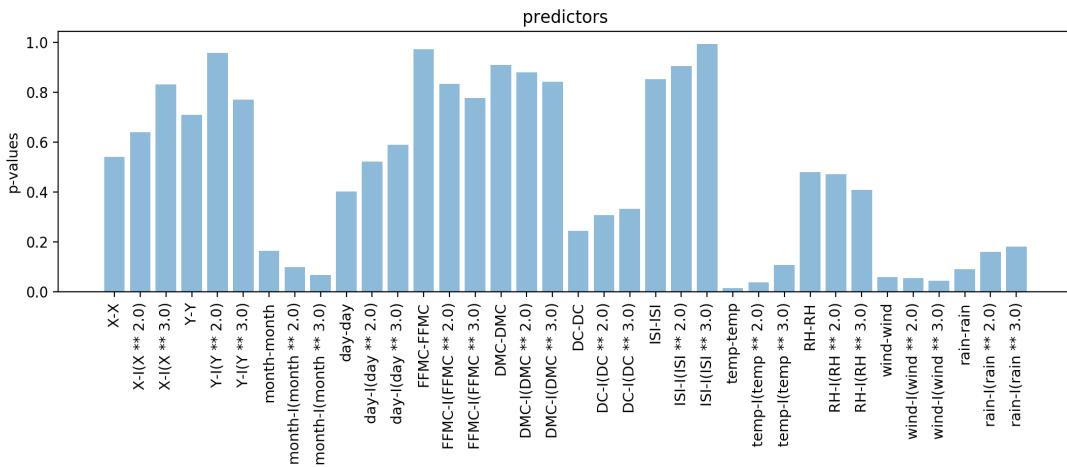
Omnibus:	95.367	Durbin-Watson:	0.929
Prob(Omnibus):	0.000	Jarque-Bera (JB):	146.689
Skew:	1.214	Prob(JB):	1.40e-32
Kurtosis:	3.955	Cond. No.	2.53e+03

Plots of coefficients of in polynomial regression:



Plot of p-values in polynomial regression:

The p-values of temperature, wind and rain have evidently improved in the model. Hence these can be used in the prediction process to better fit the data.



(g) Is there evidence of association of interactions of predictors with the response? To answer this question, run a full linear regression model with all pairwise interaction terms and state whether any interaction terms are statistically significant.

```

import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.feature_selection import RFE
from sklearn import preprocessing

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))
intercept1=[]

```

```

pvalues1 =[]
keys1=[]
intercept2=[]
pvalues2 =[]
keys2=[]
intercept3=[]
pvalues3 =[]
keys3=[]

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"] = le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"] = le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]

model = ols("area ~
X+Y+month+day+FFMC+DMC+DC+ISI+temp+RH+wind+rain+X:Y+X:month+X:day+X:FF
MC+X:DMC+X:DC+X:ISI+X:temp+X:RH+X:wind+X:rain+Y:month+Y:day+Y:FFMC+Y:DM
C+Y:DC+Y:ISI+Y:temp+Y:RH+Y:wind+Y:rain+month:day+month:FFMC+month:DMC+mont
h:DC+month:ISI+month:temp+month:RH+month:wind+month:rain+day:FFMC+day:DMC+day
:DC+day:ISI+day:temp+day:RH+day:wind+day:rain+FFMC:DMC+FFMC:DC+FFMC:ISI+FF
MC:temp+FFMC:RH+FFMC:wind+FFMC:rain+DMC:DC+DMC:ISI+DMC:temp+DMC:RH+
DMC:wind+DMC:rain+DC:ISI+DC:temp+DC:RH+DC:wind+DC:rain+ISI:temp+ISI:RH+ISI:w
ind+ISI:rain+temp:RH+temp:wind+temp:rain+RH:wind+RH:rain+wind:rain",
normalized_df).fit()
print(model.summary())

for key in model.pvalues.keys()[1:40]:
    intercept1.append(model.params[key])
    pvalues1.append(model.pvalues[key])
    keys1.append(key)
for key in model.pvalues.keys()[40: ]:
    intercept2.append(model.params[key])
    pvalues2.append(model.pvalues[key])
    keys2.append(key)

##Graphs on p-values
y_pos = range(len(keys1))
a = plt.bar(y_pos, pvalues1, align='center', alpha=0.5)
plt.xticks(y_pos, keys1, size=8, rotation=90)
plt.ylabel('p-values')

```

```

plt.title('predictors')
plt.show()

##Graphs on p-values
y_pos = range(len(keys2))
a = plt.bar(y_pos, pvalues2, align='center', alpha=0.5)
plt.xticks(y_pos, keys2, rotation='vertical')
plt.ylabel('p-values')
plt.title('predictors')
plt.show()

```

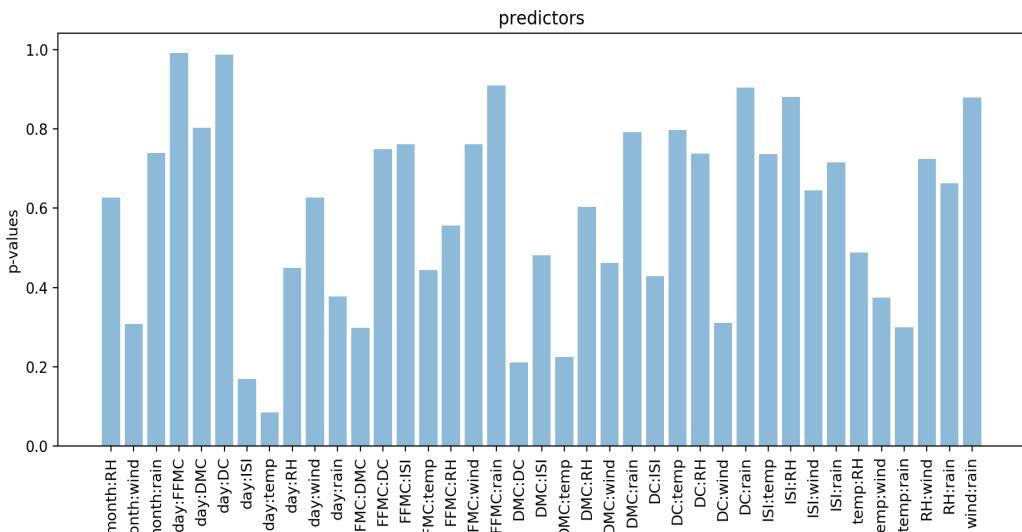
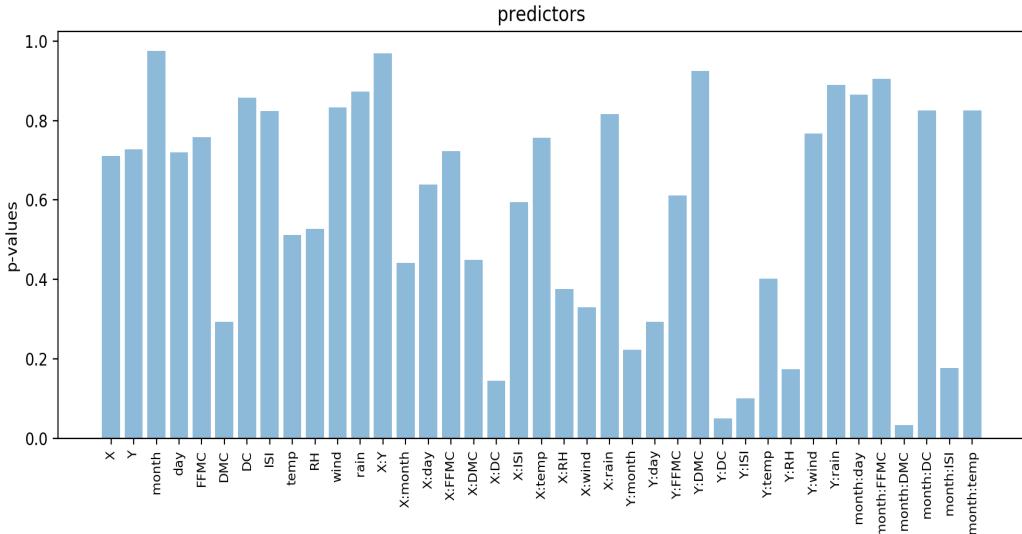
Output:

OLS Regression Results

Dep. Variable:	area	R-squared:	0.149			
Model:	OLS	Adj. R-squared:	0.007			
Method:	Least Squares	F-statistic:	1.047			
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	0.382			
Time:	02:49:22	Log-Likelihood:	-864.73			
No. Observations:	517	AIC:	1879.			
Df Residuals:	442	BIC:	2198.			
Df Model:	74					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-4.0223	16.146	-0.249	0.803	-35.754	27.709
X	-2.9855	8.066	-0.370	0.711	-18.837	12.866
Y	-5.2388	15.051	-0.348	0.728	-34.820	24.342
month	0.1734	5.559	0.031	0.975	-10.752	11.098
day	-2.4088	6.710	-0.359	0.720	-15.596	10.778
FFMC	5.9496	19.270	0.309	0.758	-31.923	43.822
DMC	-18.3697	17.443	-1.053	0.293	-52.650	15.911
DC	1.5944	8.839	0.180	0.857	-15.776	18.965
ISI	-7.4387	33.377	-0.223	0.824	-73.036	58.158
temp	14.3643	21.867	0.657	0.512	-28.612	57.341
RH	7.7378	12.250	0.632	0.528	-16.337	31.812
wind	-1.5079	7.180	-0.210	0.834	-15.620	12.604
rain	-8.0701	50.308	-0.160	0.873	-106.943	90.803
X:Y	-0.0447	1.166	-0.038	0.969	-2.337	2.248
X:month	0.6638	0.861	0.771	0.441	-1.029	2.356
X:day	0.4210	0.897	0.470	0.639	-1.341	2.183
X:FFMC	3.1701	8.954	0.354	0.723	-14.428	20.768
X:DMC	-1.5075	1.992	-0.757	0.450	-5.422	2.407
X:DC	-2.3900	1.636	-1.461	0.145	-5.606	0.826
X:ISI	2.7787	5.211	0.533	0.594	-7.463	13.020
X:temp	0.7656	2.468	0.310	0.757	-4.086	5.617
X:RH	1.9443	2.193	0.886	0.376	-2.367	6.255
X:wind	1.5352	1.573	0.976	0.330	-1.556	4.626
X:rain	26.4613	113.998	0.232	0.817	-197.584	250.507
Y:month	-1.7599	1.445	-1.218	0.224	-4.599	1.079
Y:day	1.5345	1.461	1.051	0.294	-1.336	4.405
Y:FFMC	8.9531	17.641	0.508	0.612	-25.718	43.624
Y:DMC	-0.3463	3.649	-0.095	0.924	-7.518	6.825
Y:DC	5.1893	2.646	1.961	0.051	-0.011	10.390
Y:ISI	-15.0691	9.165	-1.644	0.101	-33.082	2.944
Y:temp	-3.6514	4.357	-0.838	0.402	-12.215	4.912

Y:RH	-5.3939	3.959	-1.362	0.174	-13.175	2.387
Y:wind	0.8061	2.715	0.297	0.767	-4.529	6.141
Y:rain	-27.0072	193.547	-0.140	0.889	-407.394	353.380
month:day	0.1146	0.672	0.171	0.865	-1.205	1.434
month:FFMC	0.7367	6.165	0.119	0.905	-11.380	12.853
month:DMC	3.1904	1.493	2.137	0.033	0.256	6.125
month:DC	-0.3388	1.529	-0.222	0.825	-3.344	2.666
month:ISI	-5.5707	4.121	-1.352	0.177	-13.671	2.529
month:temp	0.4299	1.942	0.221	0.825	-3.387	4.247
month:RH	-0.8157	1.678	-0.486	0.627	-4.114	2.483
month:wind	-1.1163	1.093	-1.021	0.308	-3.264	1.032
month:rain	74.4953	223.512	0.333	0.739	-364.783	513.773
day:FFMC	0.0850	7.517	0.011	0.991	-14.689	14.859
day:DMC	0.4003	1.599	0.250	0.802	-2.741	3.542
day:DC	-0.0204	1.269	-0.016	0.987	-2.514	2.473
day:ISI	-6.3243	4.594	-1.377	0.169	-15.354	2.705
day:temp	3.5995	2.083	1.728	0.085	-0.494	7.693
day:RH	1.3418	1.774	0.756	0.450	-2.144	4.828
day:wind	0.6003	1.234	0.486	0.627	-1.826	3.026
day:rain	-28.3175	32.008	-0.885	0.377	-91.224	34.589
FFMC:DMC	19.2444	18.485	1.041	0.298	-17.085	55.573
FFMC:DC	-2.9225	9.117	-0.321	0.749	-20.841	14.996
FFMC:ISI	10.8079	35.538	0.304	0.761	-59.037	80.653
FFMC:temp	-19.5141	25.491	-0.766	0.444	-69.613	30.585
FFMC:RH	-8.4149	14.273	-0.590	0.556	-36.466	19.636
FFMC:wind	2.4433	8.027	0.304	0.761	-13.332	18.219
FFMC:rain	-7.4241	64.750	-0.115	0.909	-134.681	119.833
DMC:DC	-2.9297	2.340	-1.252	0.211	-7.529	1.669
DMC:ISI	-7.6520	10.863	-0.704	0.482	-29.001	13.697
DMC:temp	6.6374	5.460	1.216	0.225	-4.094	17.369
DMC:RH	2.3283	4.484	0.519	0.604	-6.484	11.140
DMC:wind	-2.1906	2.975	-0.736	0.462	-8.037	3.656
DMC:rain	46.9047	177.134	0.265	0.791	-301.226	395.035
DC:ISI	4.2557	5.369	0.793	0.428	-6.297	14.808
DC:temp	0.8772	3.414	0.257	0.797	-5.833	7.587
DC:RH	-0.9058	2.697	-0.336	0.737	-6.205	4.394
DC:wind	2.0537	2.022	1.016	0.310	-1.920	6.027
DC:rain	-5.3174	43.789	-0.121	0.903	-91.377	80.742
ISI:temp	4.7383	14.031	0.338	0.736	-22.838	32.315
ISI:RH	1.5195	10.088	0.151	0.880	-18.307	21.346
ISI:wind	3.2351	7.022	0.461	0.645	-10.566	17.037
ISI:rain	-78.3362	214.507	-0.365	0.715	-499.916	343.244
temp:RH	1.7306	2.491	0.695	0.488	-3.165	6.626
temp:wind	-2.8258	3.182	-0.888	0.375	-9.080	3.429
temp:rain	82.0916	79.073	1.038	0.300	-73.314	237.497
RH:wind	-0.8790	2.483	-0.354	0.724	-5.760	4.002
RH:rain	-71.8417	164.595	-0.436	0.663	-395.328	251.644
wind:rain	-13.3816	87.837	-0.152	0.879	-186.013	159.249
<hr/>						
Omnibus:	85.137	Durbin-Watson:			1.119	
Prob(Omnibus):	0.000	Jarque-Bera (JB):			127.659	
Skew:	1.086	Prob(JB):			1.90e-28	
Kurtosis:	4.098	Cond. No.			1.10e+16	
<hr/>						

Plots:



In the plots we can see that few interaction terms seem to have good correlation with data. These terms can be considered for stepwise selection to output a good model that has statistical significance.

We can improve our model by considering the significant interaction terms, non-linear terms and linear terms to create a model that would come up with the least error rate.

Significant interaction terms: day:ISI , day:temp, x:DC,Y:DC,Y:ISI,month:DMC.

The Rsquared value has also improved drastically from 0.028 for multiple linear regression to 0.149 which visibly explains the error much better than linear regression. Hence interaction terms will be vital in our model building.

(h) Can you improve your model using possible interaction terms or nonlinear associations and between the predictors and response? Train the model on a randomly selected 70% subset of the data and test it on the remaining points and report your train and test results

Note:

In the following code I perform a stepwise selection of features by running a full linear regression on all possible terms and removing the term with the highest p-value.

The process stops when the R-squared value stops improving to give the most relevant subset of features.

After getting hold of the best subset of features a linear regression is performed to predict the area of the 30% data separated from the dataset for testing.

Finally the mean-squared error is calculated.

Code:

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.model_selection import train_test_split
import itertools
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import preprocessing

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))
formulaList =
["X","Y","month","day","FFMC","DMC","DC","ISI","temp","RH","wind","rain","X:Y","X:FFMC","X :DMC","X:DC","X:ISI","X:temp","X:RH","X:wind","X:rain","Y:FFMC","Y:DMC","Y:DC","Y:ISI","Y:te mp","Y:RH","Y:wind","Y:rain","FFMC:DMC","FFMC:DC","FFMC:ISI","FFMC:temp","FFMC:RH","F FMC:wind","FFMC:rain","DMC:DC","DMC:ISI","DMC:temp","DMC:RH","DMC:wind","DMC:rain"," DC:ISI","DC:temp","DC:RH","DC:wind","DC:rain","ISI:temp","ISI:RH","ISI:wind","ISI:rain","temp :RH","temp:wind","temp:rain","RH:wind","RH:rain","wind:rain","month:day","month:FFMC"," month:DMC","month:DC","month:ISI","month:temp","month:wind","month:rain","month:RH"," day:FFMC","day:DMC","day:DC","day:ISI","day:temp","day:RH","day:wind","day:rain"]
```

```

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"] = le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"] = le.transform(ff_df["day"])

normalized_df = (ff_df.iloc[:, :12] - ff_df.iloc[:, :12].min()) / (ff_df.iloc[:, :12].max() - ff_df.iloc[:, :12].min())
normalized_df["area"] = ff_df["area"]

for column in list(normalized_df.columns[:-1]):
    formulaList.append("I(" + column + " ** 2.0)")
    formulaList.append("I(" + column + " ** 3.0)")

#print(formulaList)
bestFormula = None
bestRsquared = 100
train, test = train_test_split(normalized_df, test_size=0.3)
while (len(formulaList) > 2
      "0):
    formula_now = ""
    for formula in formulaList:
        formula_now += "+" + formula
    model = ols("area ~" + formula_now, train).fit()
    op = model.predict(test)
    test["predictedArea"] = op
    rms = np.sqrt(mean_squared_error(test["area"], test["predictedArea"]))
    pvalues = pd.DataFrame()
    pvalues = model.pvalues[1:].sort_values(ascending=False)
    # print(model.pvalues[1:].sort_values(ascending=False))
    # print(pvalues.keys()[0])
    formulaList.remove(pvalues.keys()[0])
    if(bestRsquared > rms):
        bestFormula = formula_now
        bestRsquared = rms
print(bestFormula)
print(bestRsquared)

model = ols("area ~" + bestFormula, train).fit()
print(model.summary())
op = model.predict(test)
#print(op)
test["predictedArea"] = op
print(test)

```

```

rms = np.sqrt(mean_squared_error(test["area"], test["predictedArea"]))
print(rms)

```

Output:

OLS Regression Results									
Dep. Variable:	area	R-squared:	0.158						
Model:	OLS	Adj. R-squared:	0.106						
Method:	Least Squares	F-statistic:	3.032						
Date:	Sat, 03 Feb 2018	Prob (F-statistic):	1.30e-05						
Time:	02:56:44	Log-Likelihood:	-609.56						
No. Observations:	361	AIC:	1263.						
Df Residuals:	339	BIC:	1349.						
Df Model:	21								
Covariance Type:	nonrobust								
	coef	std err	t	P> t	[0.025	0.975]			
Intercept	-6.5507	4.164	-1.573	0.117	-14.742	1.640			
DC	-1.2344	0.595	-2.074	0.039	-2.405	-0.064			
ISI	-75.8325	39.497	-1.920	0.056	-153.523	1.858			
RH	5.8985	4.483	1.316	0.189	-2.920	14.717			
X:DMC	-2.3666	0.889	-2.663	0.008	-4.115	-0.618			
X:wind	1.8648	0.717	2.599	0.010	0.454	3.276			
FFMC:ISI	80.0580	40.412	1.981	0.048	0.569	159.547			
FFMC:temp	-11.9665	3.600	-3.324	0.001	-19.048	-4.885			
FFMC:RH	-11.3437	5.615	-2.020	0.044	-22.388	-0.300			
DMC:temp	4.0467	1.213	3.336	0.001	1.661	6.433			
ISI:RH	7.5050	6.242	1.202	0.230	-4.772	19.782			
temp:RH	7.4038	2.805	2.639	0.009	1.886	12.922			
month:DC	2.3244	0.639	3.635	0.000	1.067	3.582			
month:ISI	-6.7099	2.827	-2.373	0.018	-12.271	-1.149			
day:DC	0.5750	0.348	1.651	0.100	-0.110	1.260			
day:rain	-26.6163	12.360	-2.153	0.032	-50.929	-2.304			
I(X ** 3.0)	1.1408	0.541	2.109	0.036	0.077	2.205			
I(Y ** 3.0)	-0.9207	0.654	-1.407	0.160	-2.208	0.367			
I(FFMC ** 2.0)	34.5538	17.887	1.932	0.054	-0.630	69.738			
I(FFMC ** 3.0)	-23.7935	14.143	-1.682	0.093	-51.614	4.026			
I(temp ** 2.0)	7.1288	2.604	2.738	0.007	2.008	12.250			
I(rain ** 2.0)	21.4845	10.275	2.091	0.037	1.274	41.695			
Omnibus:	68.610	Durbin-Watson:	1.624						
Prob(Omnibus):	0.000	Jarque-Bera (JB):	109.291						
Skew:	1.143	Prob(JB):	1.85e-24						
Kurtosis:	4.429	Cond. No.	1.64e+03						

	area	predictedArea
487	2.856470	0.971857
180	1.845300	1.123156
53	0.000000	0.959808
406	1.599388	1.162061
156	0.959350	1.581326
151	0.858662	0.637886
204	2.707383	0.314267
40	0.000000	0.843878
214	3.389799	0.545852
77	0.000000	0.664550
248	0.000000	1.266927
131	0.000000	0.140286

306	0.343590	1.351954
237	5.365415	0.704589
348	0.000000	0.921876
89	0.000000	0.756112
55	0.000000	1.051854
165	1.144223	1.332324
277	2.462150	2.794532
59	0.000000	0.538765
444	2.394252	2.103993
377	5.168380	1.035754
106	0.000000	0.035043
58	0.000000	0.497600
353	1.000632	1.059121
107	0.000000	1.137977
143	0.536493	0.529147
161	1.064711	1.266747
11	0.000000	0.598062
221	3.607669	0.961693
..
264	2.020222	1.018894
299	0.000000	-0.418371
447	0.000000	1.407879
441	0.802002	1.307570
63	0.000000	0.571280
169	1.266948	1.235724
108	0.000000	1.149699
105	0.000000	0.671062
216	3.417071	0.673884
80	0.000000	0.911406
342	0.000000	0.754115
230	4.494127	1.272081
271	2.329227	1.144417
47	0.000000	0.691336
403	0.000000	1.480709
197	2.489894	1.083204
300	0.000000	0.878359
440	0.000000	1.046168
269	1.686399	0.797218
65	0.000000	0.638433
357	0.000000	0.880573
135	0.000000	0.685792
485	1.121678	0.985141
303	0.000000	0.391080
66	0.000000	0.839689
258	0.000000	0.873838
111	0.000000	0.020661
316	0.000000	1.584256
469	4.129229	2.278745
378	0.000000	1.611279

[156 rows x 14 columns]

Mean – Squared error : 1.3252481839

The mean squared error achieved in this regression task is 1.325.

(i) KNN Regression: Note that for this problem, we have a mixture of categorical and quantitative predictors. There is not a unique way to define a distance metric in such a situation. Describe your findings and heuristics. Can your metric be specific to this problem? Use a reasonable distance metric to answer the following questions:

Categorical data cannot be classified using Euclidean distance. We have an approach called hamming distance used in classification algorithms to calculate distances between categorical data.

Our categorical data here have an implied ordering which can be positively used to predict area depending on which month or day they fall in. Similarly, co-ordinates also have an implied order and the order can be used positively in a KNN problem.

Another important factor to be considered is normalization of variables. Since we go about calculating feature distances between various training samples it is important that all features are normalized by their range hence, no variable dominates the other variables because of its range.

- i. **Use the first 4 predictors to perform k-nearest neighbor regression for this dataset. Find the value of k that gives you the best fit. Plot the train and test errors in terms of 1/k.**

Code:

```
import pandas as pd
import math
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.model_selection import train_test_split
import itertools
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.neighbors import KNeighborsRegressor

def hammingDistance(a,b):
    return abs(np.sum(a-b))

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"]=le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"]=le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]
```

```

print(normalized_df)

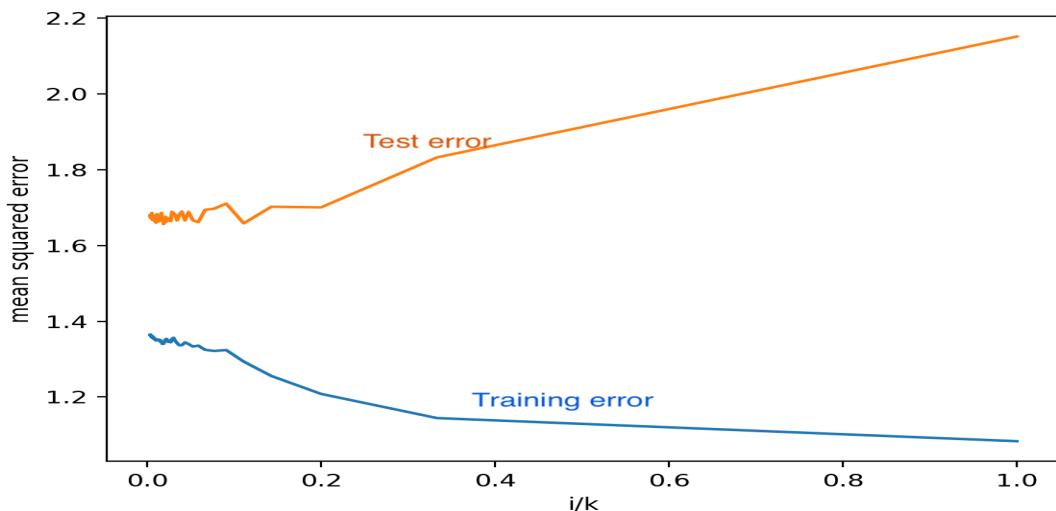
train, test = train_test_split(ff_df, test_size=0.1)

scores_training = []
scores_test = []
neighbours = []
for k in range (1,100,2):
    neighbours.append(1/k)
    print(k)
    neigh = KNeighborsRegressor(n_neighbors=k,metric=hammingDistance)
    neigh.fit(train.iloc[:,4], train["area"])
    result = neigh.predict(train.iloc[:,4])
    train["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(train["area"], train["predictedArea"]))
    scores_training.append(rms)
    print(rms)
    result = neigh.predict(test.iloc[:,4])
    test["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(test["area"], test["predictedArea"]))
    scores_test.append(rms)
    print(rms)

plt.plot(neighbours,scores_training,label="training mean squared error")
plt.plot(neighbours,scores_test,label="test mean squared error")
plt.xlabel('i/k')
plt.ylabel('mean squared error')
plt.show()

```

Output:



On analyzing the test and training errors, k^* is can be concluded to be around 83.

- ii. **Use the last 4 predictors to perform k-nearest neighbor regression for this dataset. Find the value of k that gives you the best fit. Plot the train and test errors in terms of 1/k.**

Code:

```
import pandas as pd
import math
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.model_selection import train_test_split
import itertools
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.neighbors import KNeighborsRegressor

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"]=le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"]=le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-
ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]
print(normalized_df)

train, test = train_test_split(normalized_df, test_size=0.1)

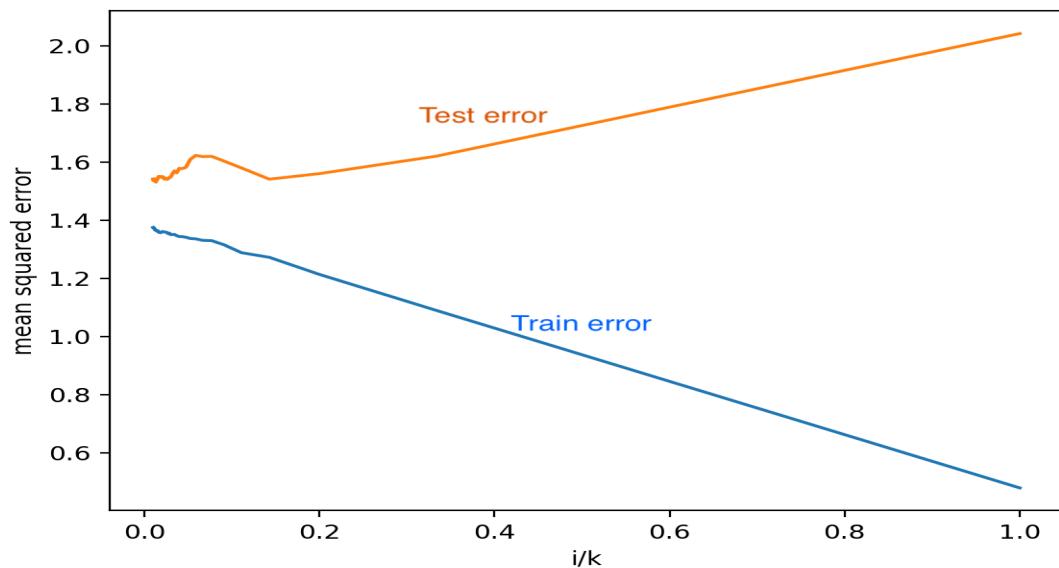
scores_training = []
scores_test = []
neighbours = []
```

```

for k in range (1,100,2):
    neigh = KNeighborsRegressor(n_neighbors=k)
    neighbours.append(1/k)
    print(k)
    neigh.fit(train.iloc[:,8:12], train["area"])
    result = neigh.predict(train.iloc[:,8:12])
    train["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(train["area"], train["predictedArea"]))
    scores_training.append(rms)
    print(rms)
    result = neigh.predict(test.iloc[:,8:12])
    test["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(test["area"], test["predictedArea"]))
    scores_test.append(rms)
    print(rms)
plt.plot(neighbours,scores_training,label="training mean squared error")
plt.plot(neighbours,scores_test,label="test mean squared error")
plt.xlabel('i/k')
plt.ylabel('mean squared error')
plt.show()

```

Output:



From the observed values of test and training error the k^* is around 45 for the last 4 predictors also.

iii. Use predictors 1,2, 9, 10, 11 to perform k-nearest neighbor regression for this dataset. Find the value of k that gives you the best fit. Plot the train and test errors in terms of 1/k.

Code:

```
import pandas as pd
import math
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.model_selection import train_test_split
import itertools
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.neighbors import KNeighborsRegressor

def hammingDistance(a,b):
    return abs(np.sum(a-b))

ff_df = pd.read_csv("forestfires.csv")
ff_df['area']=ff_df['area'].apply(lambda x:math.log1p(x))

le = preprocessing.LabelEncoder()
le.fit(ff_df["month"])
ff_df["month"]=le.transform(ff_df["month"])
le.fit(ff_df["day"])
ff_df["day"]=le.transform(ff_df["day"])

normalized_df=(ff_df.iloc[:,12]-ff_df.iloc[:,12].min())/(ff_df.iloc[:,12].max()-ff_df.iloc[:,12].min())
normalized_df["area"] = ff_df["area"]
#print(normalized_df)
final_df = pd.DataFrame()

final_df=pd.concat([normalized_df.iloc[:,0:2], normalized_df.iloc[:,8:11]], axis=1)
final_df["area"]=ff_df["area"]
print(final_df)
train, test = train_test_split(final_df, test_size=0.1)

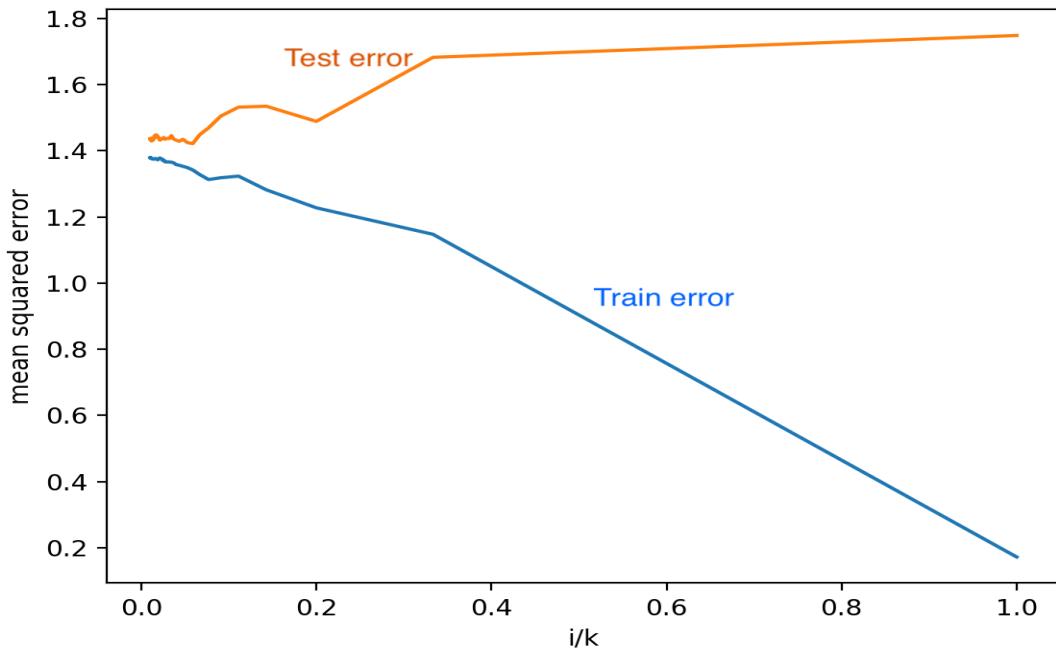
scores_training = []
scores_test = []
neighbours = []
```

```

for k in range (1,100,2):
    neighbours.append(1/k)
    print(k)
    neigh = KNeighborsRegressor(n_neighbors=k)
    neigh.fit(train.iloc[:,0:5], train["area"])
    result = neigh.predict(train.iloc[:,0:5])
    train["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(train["area"], train["predictedArea"]))
    scores_training.append(rms)
    print(rms)
    result = neigh.predict(test.iloc[:,0:5])
    test["predictedArea"] = result
    rms = np.sqrt(mean_squared_error(test["area"], test["predictedArea"]))
    print(rms)
    scores_test.append(rms)
plt.plot(neighbours,scores_training,label="training mean squared error")
plt.plot(neighbours,scores_test,label="test mean squared error")
plt.xlabel('i/k')
plt.ylabel('mean squared error')
plt.show()

```

Output:



$k^* = 47$ (from the test error results)

(j) Compare the results of KNN Regression with linear regression and provide your analysis.

The mean squared error in Linear regression: 1.3252481839

The mean squared error in knn regression:

First 4 predictors: 1.66

Last 4 predictors: 1.52

Predictors 1,2,9, and 10: 1.433

Hence comparative analysis shows better performance of linear regression with stepwise selection performing better over knn regression of selected predictors.