In [1]:
```
import warnings
warnings.filterwarnings("ignore")
from matplotlib import pyplot as plt
#from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
import pandas as pd
from sklearn.model_selection import KFold, train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
import numpy as np

from sklearn.cross_validation import cross_val_score, cross_val_predict
from sklearn import metrics
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41: De
precationWarning: This module was deprecated in version 0.18 in favor of
the model_selection module into which all the refactored classes and func
tions are moved. Also note that the interface of the new CV iterators are
different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:
```
def change_month(month):
    month = month.lower()

    month = month.replace('jan','1')
    month = month.replace('feb','2')
    month = month.replace('mar','3')
    month = month.replace('apr','4')
    month = month.replace('may','5')
    month = month.replace('jun','6')
    month = month.replace('jul','7')
    month = month.replace('aug','8')
    month = month.replace('sep','9')
    month = month.replace('oct','10')
    month = month.replace('nov','11')
    month = month.replace('dec','12')

    return month

def change_day(day):
    day = day.lower()

    day = day.replace('mon','1')
    day = day.replace('tue','2')
    day = day.replace('wed','3')
    day = day.replace('thu','4')
    day = day.replace('fri','5')
    day = day.replace('sat','6')
    day = day.replace('sun','7')

    return day
```

In [3]:
```
df = pd.read_csv('forestfires.csv')
# Get names of indexes for which column Age has value 30
indexNames = df[ df['area'] <= 0].index

# Delete these row indexes from dataFrame
df.drop(indexNames , inplace=True)
indexNames = df[df['area'] >400].index
df.drop(indexNames,inplace=True)

df['month'] = df['month'].apply(change_month)
df['day'] = df['day'].apply(change_day)

Y = df['area'].values
X1 = df['FFMC'].values
X2 = df['DMC'].values
X3 = df['DC'].values
X4 = df['ISI'].values
X5 = df['temp'].values
X6 = df['RH'].values
X7 = df['wind'].values
X8 = df['rain'].values
print(np.mean(Y))
#Create combined X matrix, which will be used to predict Price
ones = np.ones((268,1))
X_all = np.column_stack((ones,X1,X2,X3,X4,X5,X6,X7,X8))
lm = LinearRegression()
lm.fit(X_all, Y)
```

```
17.9288432836
```

Out[3]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
se)
```

Only Ones Column M0 (Null Hypothesis)

In [4]:
```
from sklearn.model_selection import KFold # import KFold
X =  ones
#scaler = MinMaxScaler(feature_range=(0,1))
#X = scaler.fit_transform(X)
kf = KFold(n_splits=5,shuffle=True) # Define the split - into 2 folds
k_amount = kf.get_n_splits(X) # returns the number of splitting iterations
```

In [5]:
```
fig, axes = plt.subplots(3, 2,figsize=(15,15))
plot =[axes[0,0],axes[0,1],axes[1,0],axes[1,1],axes[2,0],axes[2,1]]
iter_plot = 0
total_train_SSE = 0
total_test_SSE = 0
total_w = 0
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]
    #Find the weight by using train set
    w_train = np.linalg.lstsq(X_train,Y_train)[0]
    total_w = total_w+w_train

    #Predict the price with train and test set
    pred_train = np.dot(X_train,w_train)
    pred_test = np.dot(X_test,w_train)

    #Calculate SSE for training set and Test set
    train_SSE = sum((pred_train-Y_train)**2)
    test_SSE = sum((pred_test-Y_test)**2)
    total_train_SSE = total_train_SSE + train_SSE
    total_test_SSE = total_test_SSE + test_SSE
    plot[iter_plot].set_title(str(iter_plot+1)+' Fold')
    plot[iter_plot].bar('Train Set',train_SSE, color='k')
    plot[iter_plot].bar('Test Set',test_SSE, color='g')
    plot[iter_plot].set_ylabel('SSE')
    iter_plot = iter_plot+1

avg_train_SSE = total_train_SSE/k_amount
avg_test_SSE = total_test_SSE/k_amount
avg_w = total_w/k_amount
plot[iter_plot].set_title('AVG of Train SSE vs Test SSE')
plot[iter_plot].bar('Train Set',avg_train_SSE, color='b')
plot[iter_plot].bar('Test Set',avg_test_SSE, color='r')
plot[iter_plot].set_ylabel('SSE')
print('AVG Train SSE: ',avg_train_SSE,' AVG Test SSE: ',avg_test_SSE)
print()
print("Model M0: Area = {:.2f}".format(float(avg_w[0])))
```

```
AVG Train SSE:  274366.157632  AVG Test SSE:  69044.7159938

Model M0: Area = 17.93
```

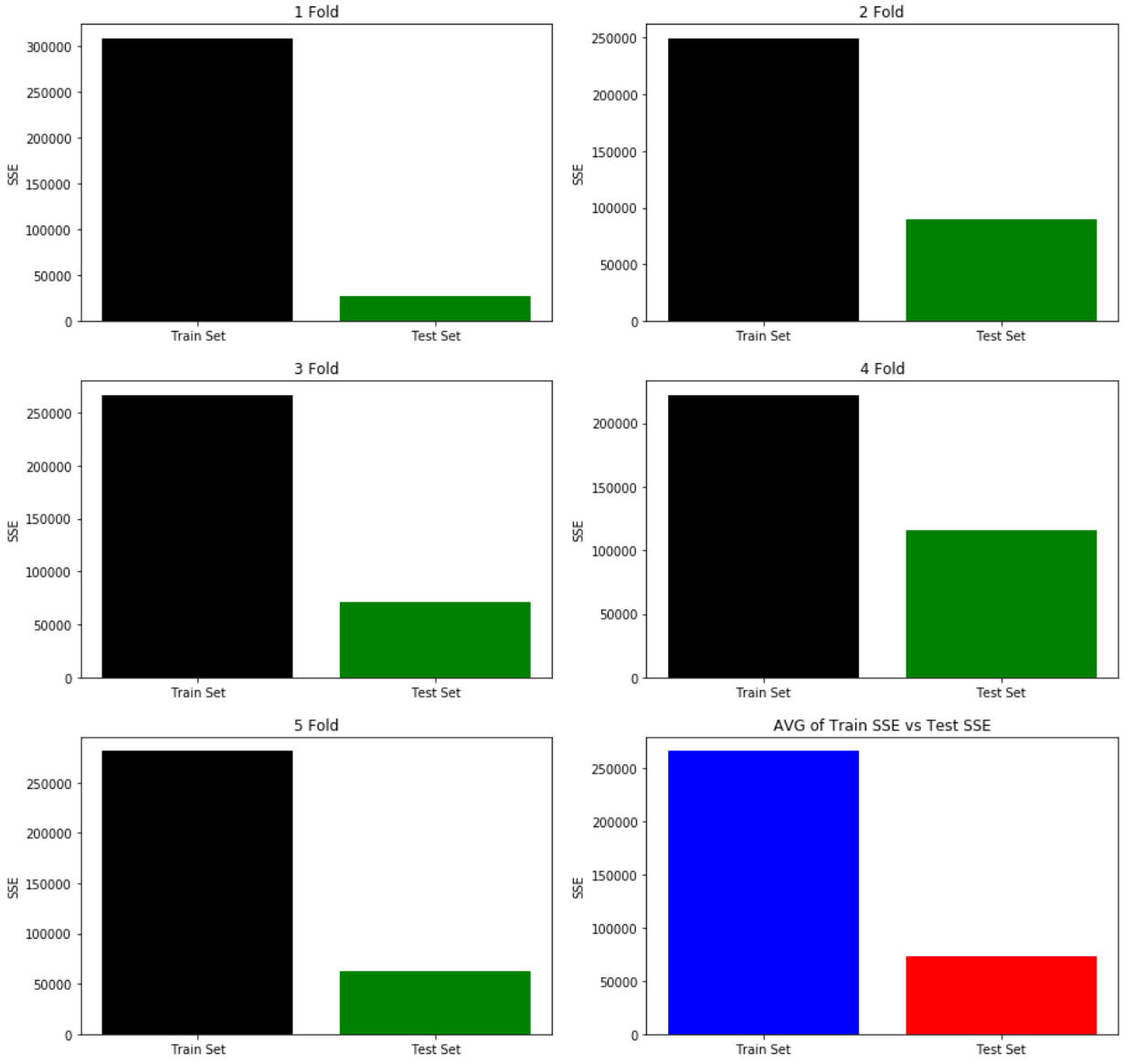
1 Fold 2 Fold

With All Variables to Predict Burned Area M1

In [6]:
```
from sklearn.model_selection import KFold # import KFold

X =  X_all
#scaler = MinMaxScaler(feature_range=(0,1))
#X = scaler.fit_transform(X)
kf = KFold(n_splits=5,shuffle=True) # Define the split - into 2 folds
```

In [7]:
```
fig, axes = plt.subplots(3, 2,figsize=(15,15))
plot =[axes[0,0],axes[0,1],axes[1,0],axes[1,1],axes[2,0],axes[2,1]]
iter_plot = 0
total_train_SSE = 0
total_test_SSE = 0
total_w = 0
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]
    #Find the weight by using train set
    w_train = np.linalg.lstsq(X_train,Y_train)[0]
    total_w = total_w+w_train

    #Predict the price with train and test set
    pred_train = np.dot(X_train,w_train)
    pred_test = np.dot(X_test,w_train)

    #Calculate SSE for training set and Test set
    train_SSE = sum((pred_train-Y_train)**2)
    test_SSE = sum((pred_test-Y_test)**2)
    total_train_SSE = total_train_SSE + train_SSE
    total_test_SSE = total_test_SSE + test_SSE
    plot[iter_plot].set_title(str(iter_plot+1)+' Fold')
    plot[iter_plot].bar('Train Set',train_SSE, color='k')
    plot[iter_plot].bar('Test Set',test_SSE, color='g')
    plot[iter_plot].set_ylabel('SSE')
    iter_plot = iter_plot+1

avg_train_SSE = total_train_SSE/k_amount
avg_test_SSE = total_test_SSE/k_amount
avg_w = total_w/k_amount
plot[iter_plot].set_title('AVG of Train SSE vs Test SSE')
plot[iter_plot].bar('Train Set',avg_train_SSE, color='b')
plot[iter_plot].bar('Test Set',avg_test_SSE, color='r')
plot[iter_plot].set_ylabel('SSE')
print('AVG Train SSE: ',avg_train_SSE,' AVG Test SSE: ',avg_test_SSE)
print()
print("Model M1: Area = {:.2f} + {:.2f} * FFMC + {:.2f} * DMC + {:.2f} * DC
 + {:.2f} * temp + {:.2f} * RH + {:.2f} * wind +{:.2f} * rain"\
      .format(float(avg_w[0]), float(avg_w[1]), float(avg_w[2]), float(avg_
              float(avg_w[4]), float(avg_w[5]),float(avg_w[6]),float(avg_w[
```

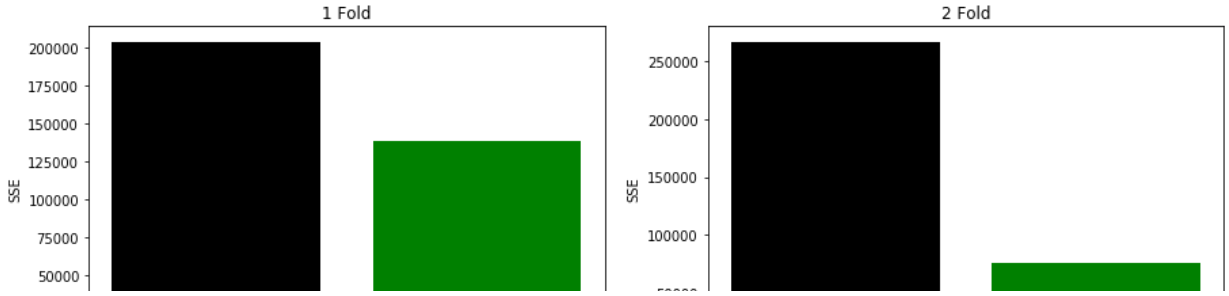
```
AVG Train SSE:  265724.336723  AVG Test SSE:  73258.4453942

Model M1: Area = -36.92 + 0.62 * FFMC + 0.06 * DMC + -0.01 * DC + -1.52 *
ISI + 0.54 * temp + -0.04 * RH + 0.67 * wind +-4.78 * rain
```



With DMC and DC to predict Burned Area M2 (Highest Positive Correlation)

In [8]:
```
from sklearn.model_selection import KFold # import KFold

X =  np.column_stack((ones,X2,X3))# create an array
#scaler = MinMaxScaler(feature_range=(0,1))
#X = scaler.fit_transform(X)
kf = KFold(n_splits=5,shuffle=True) # Define the split - into 2 folds
```

In [9]:
```
fig, axes = plt.subplots(3, 2,figsize=(15,15))
plot =[axes[0,0],axes[0,1],axes[1,0],axes[1,1],axes[2,0],axes[2,1]]
iter_plot = 0
total_train_SSE = 0
total_test_SSE = 0
total_w = 0
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]
    #Find the weight by using train set
    w_train = np.linalg.lstsq(X_train,Y_train)[0]
    total_w = total_w+w_train

    #Predict the price with train and test set
    pred_train = np.dot(X_train,w_train)
    pred_test = np.dot(X_test,w_train)

    #Calculate SSE for training set and Test set
    train_SSE = sum((pred_train-Y_train)**2)
    test_SSE = sum((pred_test-Y_test)**2)
    total_train_SSE = total_train_SSE + train_SSE
    total_test_SSE = total_test_SSE + test_SSE
    plot[iter_plot].set_title(str(iter_plot+1)+' Fold')
    plot[iter_plot].bar('Train Set',train_SSE, color='k')
    plot[iter_plot].bar('Test Set',test_SSE, color='g')
    plot[iter_plot].set_ylabel('SSE')
    iter_plot = iter_plot+1

avg_train_SSE = total_train_SSE/k_amount
avg_test_SSE = total_test_SSE/k_amount
avg_w = total_w/k_amount
plot[iter_plot].set_title('AVG of Train SSE vs Test SSE')
plot[iter_plot].bar('Train Set',avg_train_SSE, color='b')
plot[iter_plot].bar('Test Set',avg_test_SSE, color='r')
plot[iter_plot].set_ylabel('SSE')
print('AVG Train SSE: ',avg_train_SSE,' AVG Test SSE: ',avg_test_SSE)
print()
print("Model M2: Area = {:.2f} + {:.2f} * DMC + {:.2f} * DC".format(float(a
```

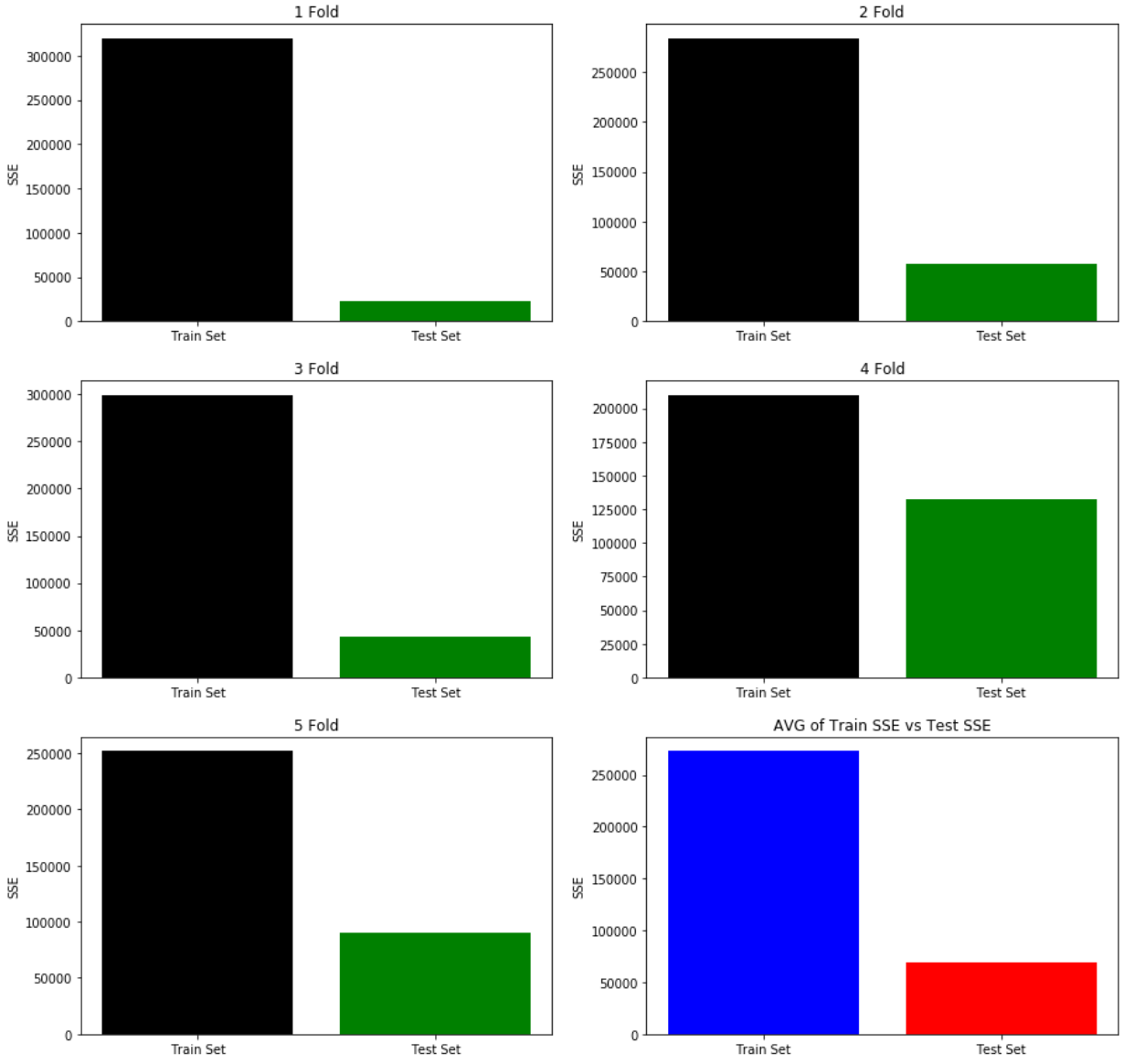


With RH and temp to predict Burned Area M3 (Highest Negative Correlation)

In [10]:
```
from sklearn.model_selection import KFold # import KFold

X =  np.column_stack((ones,X5,X6))# create an array
#scaler = MinMaxScaler(feature_range=(0,1))
#X = scaler.fit_transform(X)
kf = KFold(n_splits=5,shuffle=True) # Define the split - into 2 folds
```

In [11]:
```
fig, axes = plt.subplots(3, 2,figsize=(15,15))
plot =[axes[0,0],axes[0,1],axes[1,0],axes[1,1],axes[2,0],axes[2,1]]
iter_plot = 0
total_train_SSE = 0
total_test_SSE = 0
total_w = 0
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]
    #Find the weight by using train set
    w_train = np.linalg.lstsq(X_train,Y_train)[0]
    total_w = total_w+w_train

    #Predict the price with train and test set
    pred_train = np.dot(X_train,w_train)
    pred_test = np.dot(X_test,w_train)

    #Calculate SSE for training set and Test set
    train_SSE = sum((pred_train-Y_train)**2)
    test_SSE = sum((pred_test-Y_test)**2)
    total_train_SSE = total_train_SSE + train_SSE
    total_test_SSE = total_test_SSE + test_SSE
    plot[iter_plot].set_title(str(iter_plot+1)+' Fold')
    plot[iter_plot].bar('Train Set',train_SSE, color='k')
    plot[iter_plot].bar('Test Set',test_SSE, color='g')
    plot[iter_plot].set_ylabel('SSE')
    iter_plot = iter_plot+1

avg_train_SSE = total_train_SSE/k_amount
avg_test_SSE = total_test_SSE/k_amount
avg_w = total_w/k_amount
plot[iter_plot].set_title('AVG of Train SSE vs Test SSE')
plot[iter_plot].bar('Train Set',avg_train_SSE, color='b')
plot[iter_plot].bar('Test Set',avg_test_SSE, color='r')
plot[iter_plot].set_ylabel('SSE')
print('AVG Train SSE: ',avg_train_SSE,' AVG Test SSE: ',avg_test_SSE)
print()
print("Model M3: Area = {:.2f} + {:.2f} * RH + {:.2f} * temp".format(float(
```

```
AVG Train SSE:  272983.458753  AVG Test SSE:  69270.8423442

Model M3: Area = 14.24 + 0.30 * RH + -0.05 * temp
```

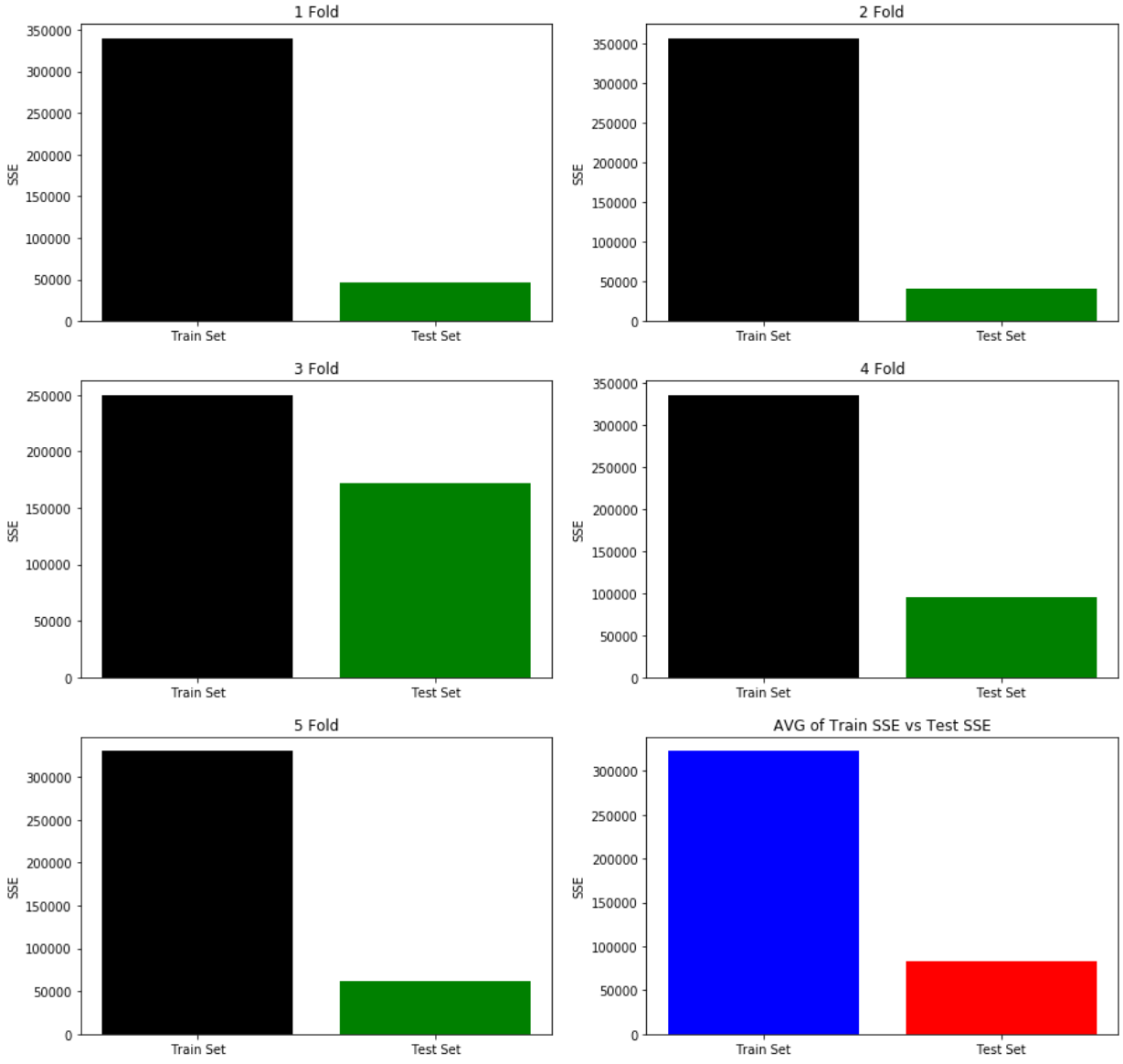


Multipling Correlated Variables to predict Burned Area M4

In [12]:
```
df['FFMC * ISI'] = np.array(list(df['FFMC'])) * np.array(list(df['ISI']))
df['FFMC * temp'] = np.array(list(df['FFMC'])) * np.array(list(df['temp']))
df['DMC * DC'] = np.array(list(df['DMC'])) * np.array(list(df['DC']))
df['DMC * temp'] = np.array(list(df['DMC'])) * np.array(list(df['temp']))
df['DC * temp'] = np.array(list(df['DC'])) * np.array(list(df['temp']))
df['temp * RH'] = np.array(list(df['temp'])) * np.array(list(df['RH']))

X = df.iloc[:, [13,14,15,16,17,18]]
y = df.iloc[:, 12]
ones = np.ones((268,1))
X = np.column_stack((ones,X))

#scaler = MinMaxScaler(feature_range=(0,1))
#X = scaler.fit_transform(X)
y = list(y)
y = np.array(y)
```

In [13]:
```
scores = []
cv = KFold(n_splits=5, shuffle=True)
k_amount = cv.get_n_splits(X)
lm = LinearRegression()
fig, axes = plt.subplots(3, 2,figsize=(15,15))
plot =[axes[0,0],axes[0,1],axes[1,0],axes[1,1],axes[2,0],axes[2,1]]
iter_plot = 0
total_train_SSE = 0
total_test_SSE = 0
total_w = 0

for train_index, test_index in cv.split(X):
    X_train = X[train_index]
    X_test = X[test_index]
    Y_train = y[train_index]
    Y_test = y[test_index]
    lm.fit(X_train, Y_train)
    w= lm.coef_
    pred_train = np.dot(X_train,w)
    pred_test = np.dot(X_test,w)
    total_w = total_w+w
    #Calculate SSE for training set and Test set
    train_SSE = sum((pred_train-Y_train)**2)
    test_SSE = sum((pred_test-Y_test)**2)
    total_train_SSE = total_train_SSE + train_SSE
    total_test_SSE = total_test_SSE + test_SSE
    plot[iter_plot].set_title(str(iter_plot+1)+' Fold')
    plot[iter_plot].bar('Train Set',train_SSE, color='k')
    plot[iter_plot].bar('Test Set',test_SSE, color='g')
    plot[iter_plot].set_ylabel('SSE')
    iter_plot = iter_plot+1
    scores.append(lm.score(X_test, Y_test))

avg_train_SSE = total_train_SSE/k_amount
avg_test_SSE = total_test_SSE/k_amount
avg_w = total_w/k_amount
plot[iter_plot].set_title('AVG of Train SSE vs Test SSE')
plot[iter_plot].bar('Train Set',avg_train_SSE, color='b')
plot[iter_plot].bar('Test Set',avg_test_SSE, color='r')
plot[iter_plot].set_ylabel('SSE')
print('AVG Train SSE: ',avg_train_SSE,' AVG Test SSE: ',avg_test_SSE)
print()
print("Model M4: Area = {:.2f} + {:.2f} * FFMC*ISI + {:.2f} * FFMC*temp + {
+ {:.2f} * DMC*temp + {:.2f} * DC*temp + {:.2f} * temp*RH "\
      .format(float(avg_w[0]),float(avg_w[1]),float(avg_w[2]),float(avg_w[3
              ,float(avg_w[5]),float(avg_w[6])))
```

```
AVG Train SSE:  322655.163145  AVG Test SSE:  83549.0283639

Model M4: Area = 0.00 + -0.01 * FFMC*ISI + 0.01 * FFMC*temp + -0.00 * DMC
*DC+ 0.00 * DMC*temp + -0.00 * DC*temp + 0.00 * temp*RH
```


1 Fold
2 Fold
3 Fold
4 Fold
5 Fold
AVG of Train SSE vs Test SSE
SSE
Train Set
Test Set

In [ ]:

In [ ]: