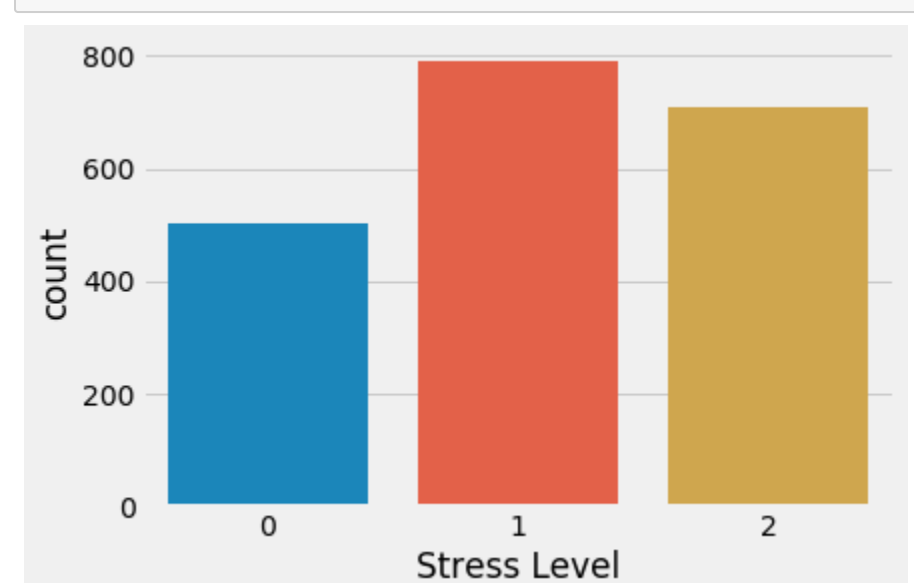
In [3]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import itertools
plt.style.use('fivethirtyeight')
```

In [5]:
```python
data=pd.read_csv('E:\ML Project\Stress-Lysis.csv')
```

In [6]:
```python
data.isnull().sum()
```

Out[6]:
```
Humidity       0
Temperature    0
Step count     0
Stress Level   0
dtype: int64
```

In [7]:
```python
sns.countplot(data["Stress Level"])
plt.show()
```



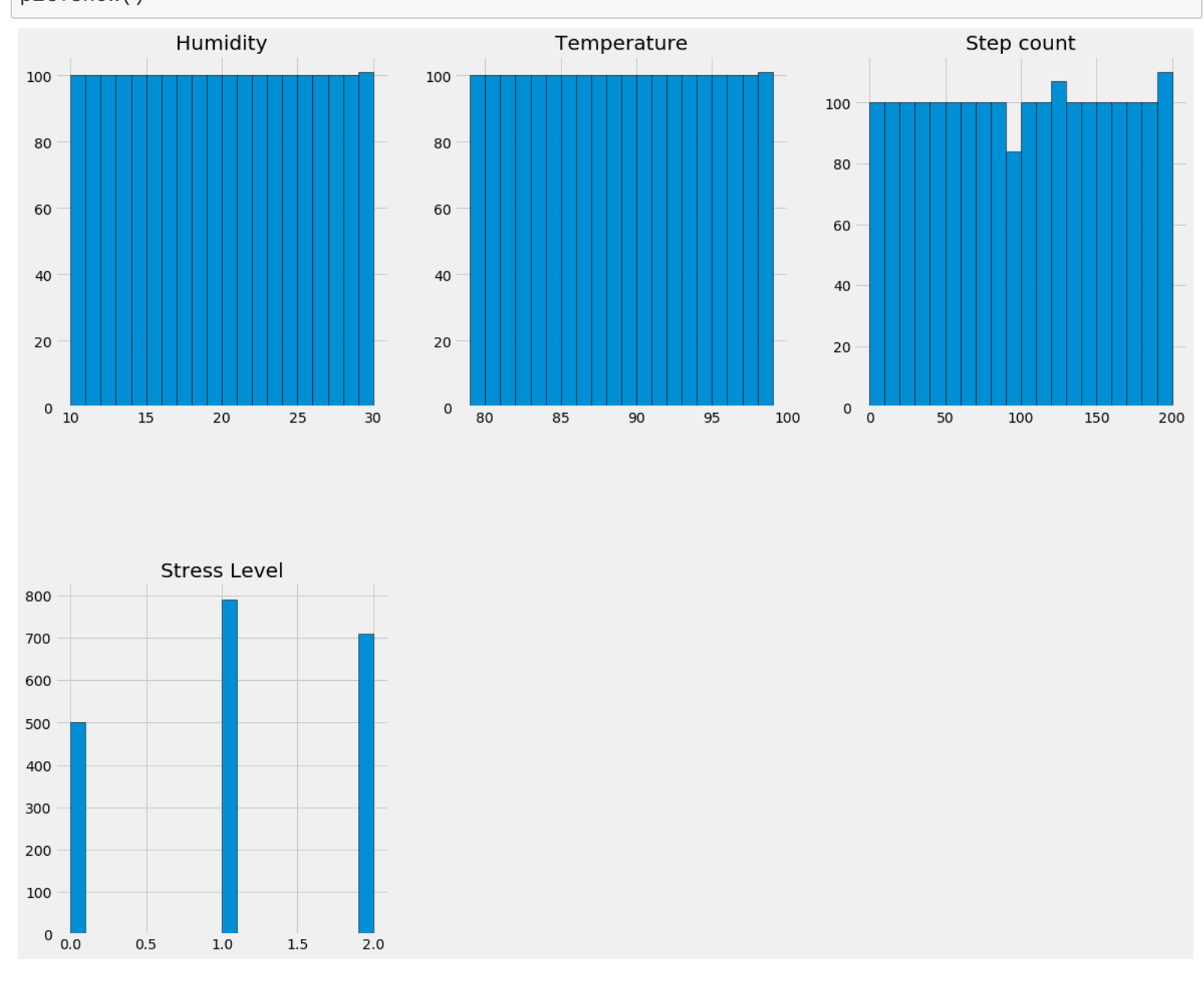
In [8]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2001 entries, 0 to 2000
Data columns (total 4 columns):
Humidity       2001 non-null float64
Temperature    2001 non-null float64
Step count     2001 non-null int64
Stress Level   2001 non-null int64
dtypes: float64(2), int64(2)
memory usage: 62.6 KB
```

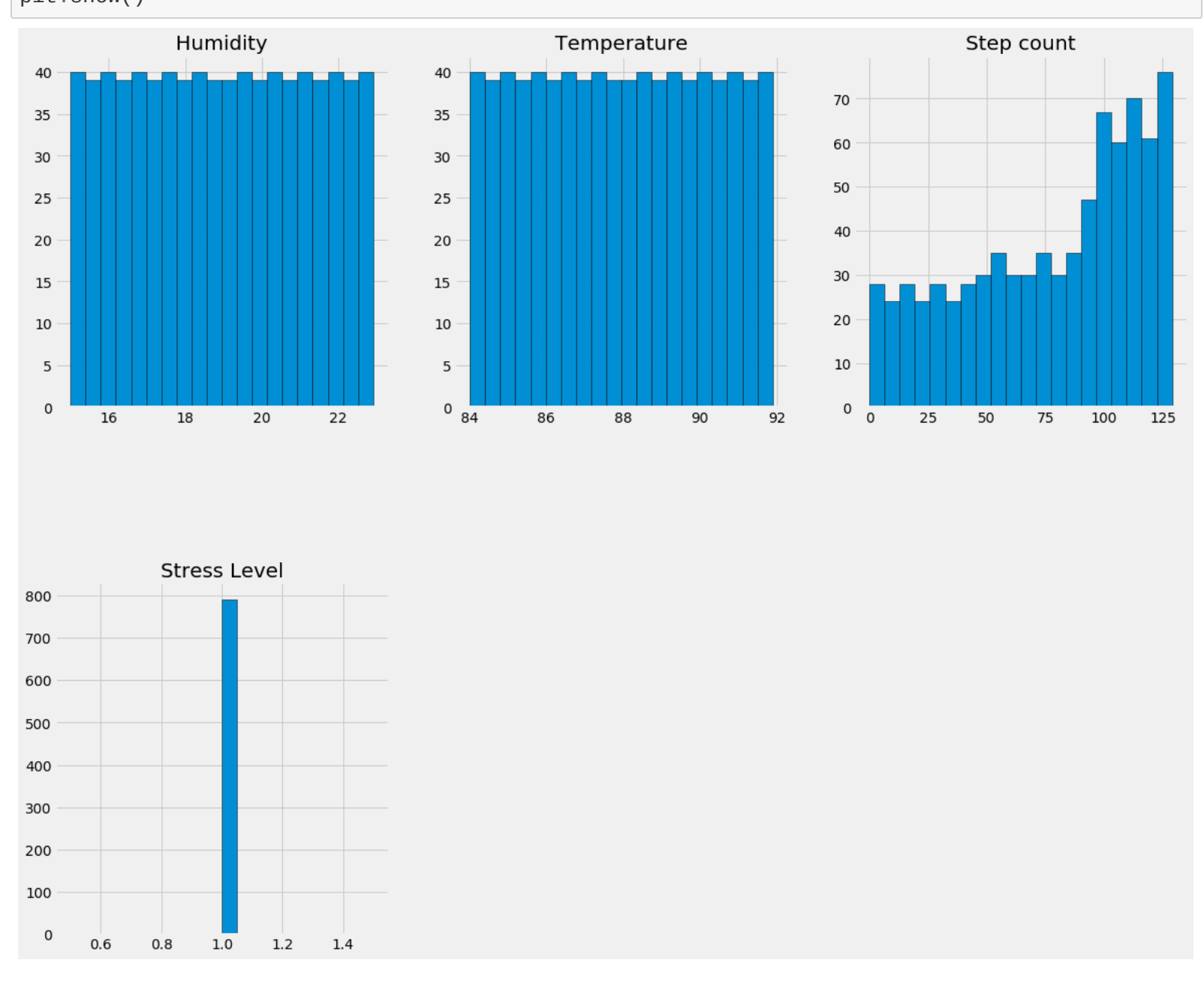
In [9]:
```python
data.columns
```

Out[9]:
```
Index(['Humidity', 'Temperature', 'Step count', 'Stress Level'], dtype='object')
```

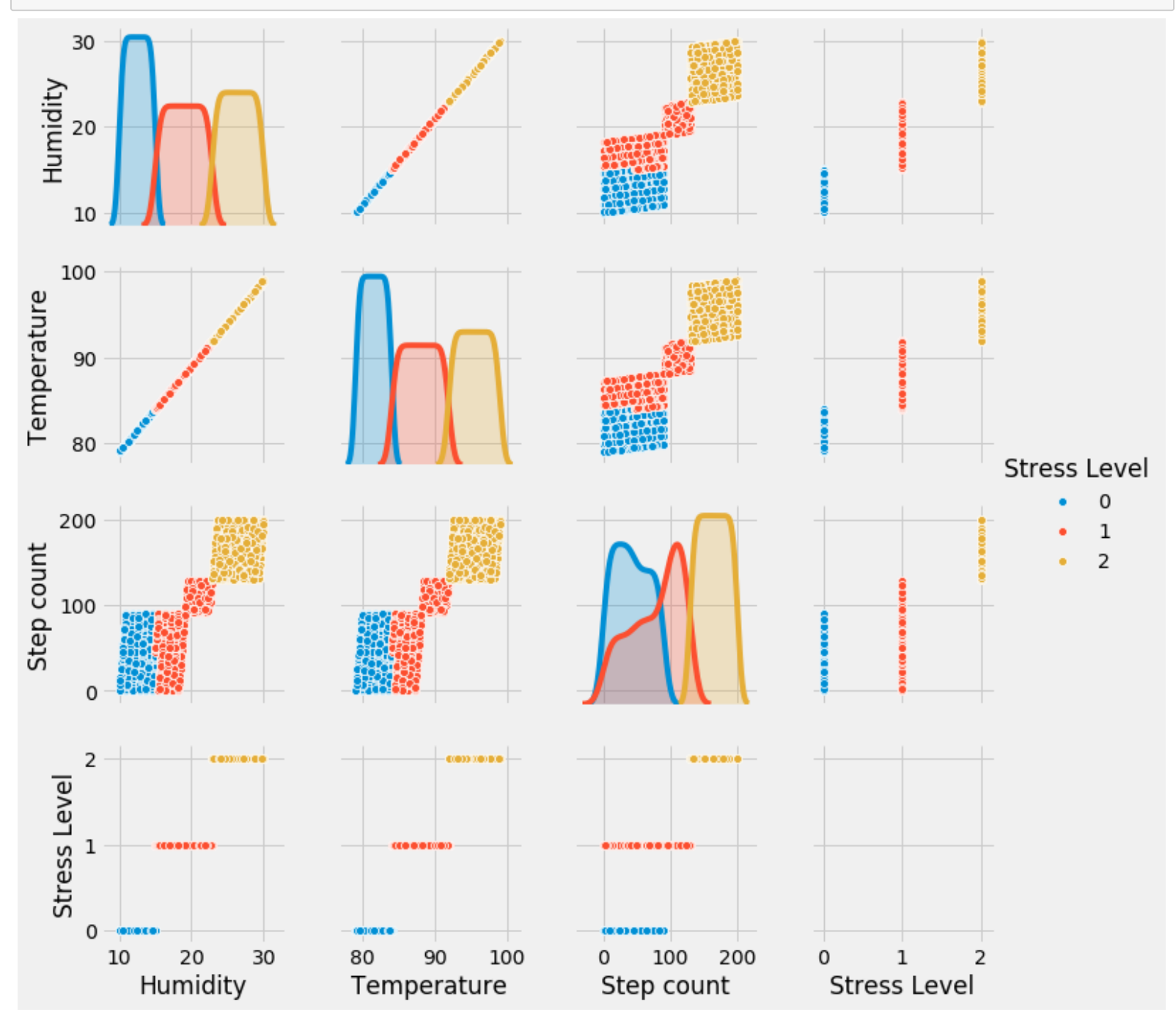
In [10]:
```python
columns=data.columns[:8]
plt.subplots(figsize=(18,15))
length=len(columns)
for i,j in itertools.zip_longest(columns,range(length)):
    plt.subplot((length/2),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    data[i].hist(bins=20,edgecolor='black')
    plt.title(i)
plt.show()
```



In [11]:
```python
# Analysis of human stress cases
data1=data[data["Stress Level"]==1]
columns=data1.columns[:8]
plt.subplots(figsize=(18,15))
length=len(columns)
for i,j in itertools.zip_longest(columns,range(length)):
    plt.subplot((length/2),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    data1[i].hist(bins=20,edgecolor='black')
    plt.title(i)
plt.show()
```



In [16]:
```python
#PairPlots:
#Lets us see the distribution of the features in the dataset
sns.pairplot(data=data,hue='Stress Level',diag_kind='kde')
plt.show()
```



In [17]:
```python
#predictive Modeling
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

In [18]:
```python
outcome=data['Stress Level']
data1=data[data.columns[:8]]
train,test=train_test_split(data,test_size=0.25,random_state=0,stratify=data['Stress Level'
])# stratify the outcome
train_X=train[train.columns[:8]]
test_X=test[test.columns[:8]]
train_Y=train['Stress Level']
test_Y=test['Stress Level']
```

In [19]:
```python
train_X.head(2)
```

Out[19]:

	Humidity	Temperature	Step count	Stress Level
1337	15.59	84.59	13	1
1448	11.85	80.85	3	0

In [20]:
```python
train_Y.head(2)
```

Out[20]:
```
1337    1
1448    0
Name: Stress Level, dtype: int64
```

In [21]:
```python
#SVM
types=['rbf','linear']
for i in types:
    model=svm.SVC(kernel=i)
    model.fit(train_X,train_Y)
    prediction=model.predict(test_X)
    print('Accuracy for SVM kernel=',i,'is',metrics.accuracy_score(prediction,test_Y))
```

```
Accuracy for SVM kernel= rbf is 1.0
Accuracy for SVM kernel= linear is 1.0
```

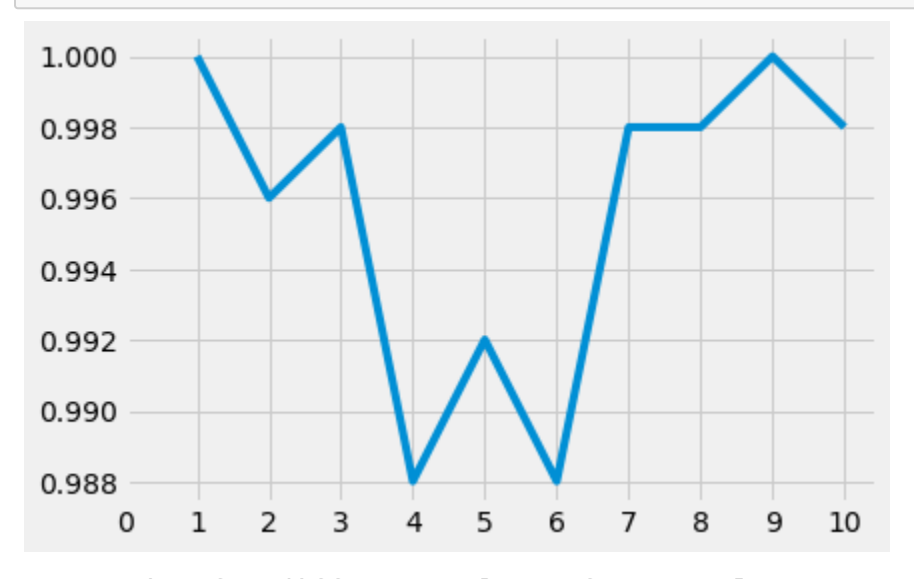
In [22]:
```python
#logistic regression
model = LogisticRegression()
model.fit(train_X,train_Y)
prediction=model.predict(test_X)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(prediction,test_Y
))
```

```
The accuracy of the Logistic Regression is 1.0
```

In [23]:
```python
#Decision Tree
model=DecisionTreeClassifier()
model.fit(train_X,train_Y)
prediction=model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(prediction,test_Y))
```

```
The accuracy of the Decision Tree is 1.0
```

In [24]:
```python
#K-Nearest Neighbours
a_index=list(range(1,11))
a=pd.Series()
x=[0,1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,11)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(train_X,train_Y)
    prediction=model.predict(test_X)
    a=a.append(pd.Series(metrics.accuracy_score(prediction,test_Y)))
plt.plot(a_index, a)
plt.xticks(x)
plt.show()
print('Accuracies for different values of n are:',a.values)
```



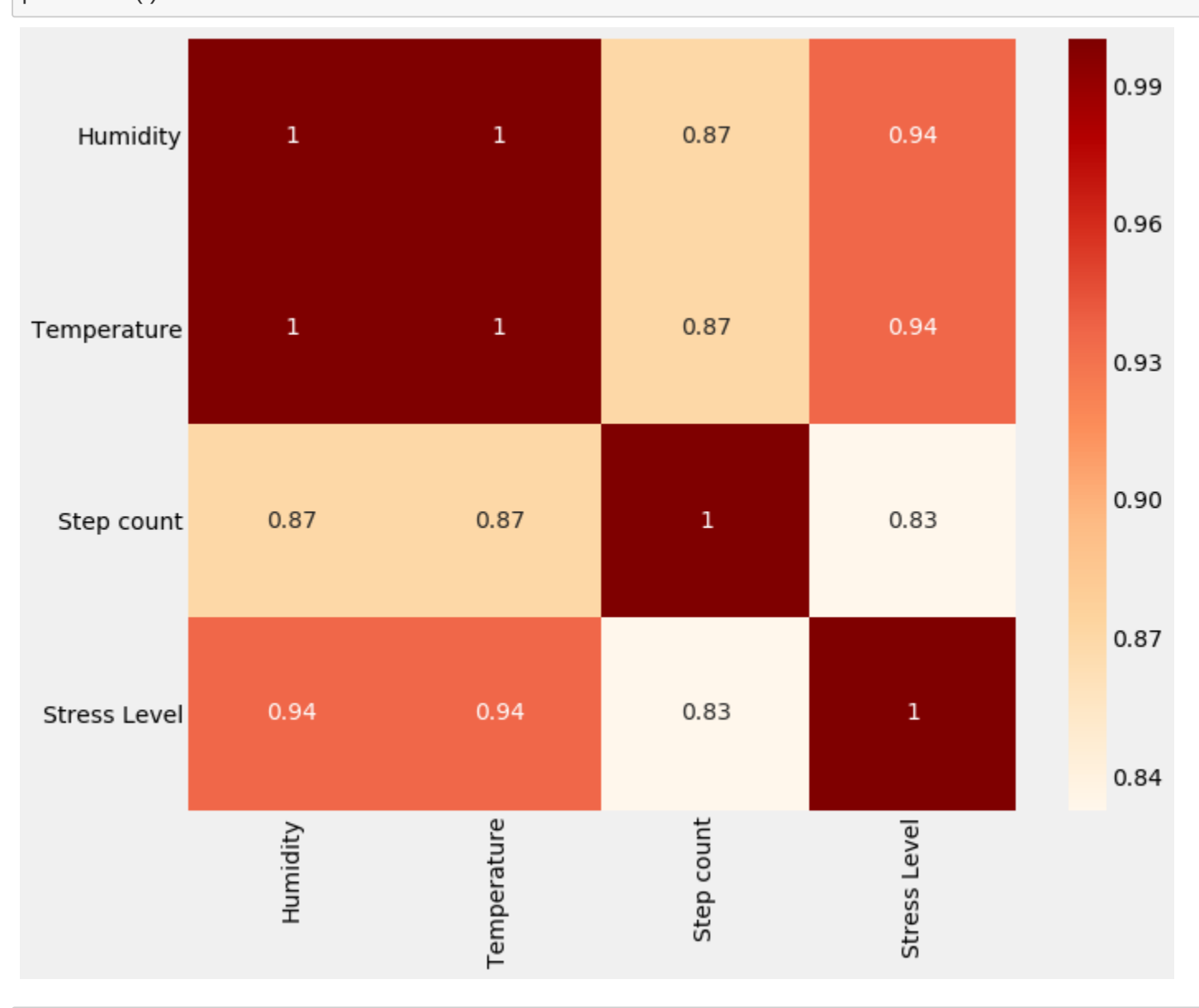
```
Accuracies for different values of n are: [1.         0.99600798 0.99800399 0.98802395 0.9920
1597 0.98802395
 0.99800399 0.99800399 1.         0.99800399]
```

In [25]:
```python
abc=[]
classifiers=['Linear Svm','Radial Svm','Logistic Regression','KNN','Decision Tree']
models=[svm.SVC(kernel='linear'),svm.SVC(kernel='rbf'),LogisticRegression(),KNeighborsClassi
fier(n_neighbors=3),DecisionTreeClassifier()]
for i in models:
    model = i
    model.fit(train_X,train_Y)
    prediction=model.predict(test_X)
    abc.append(metrics.accuracy_score(prediction,test_Y))
models_dataframe=pd.DataFrame(abc,index=classifiers)
models_dataframe.columns=['Accuracy']
models_dataframe
```

Out[25]:

	Accuracy
Linear Svm	1.000000
Radial Svm	1.000000
Logistic Regression	1.000000
KNN	0.998004
Decision Tree	1.000000

In [26]:
```python
#Correlation Matrix
sns.heatmap(data[data.columns[:8]].corr(),annot=True,cmap='OrRd')
fig=plt.gcf()
fig.set_size_inches(10,8)
plt.show()
```



In [27]:
```python
from sklearn.ensemble import RandomForestClassifier
model= RandomForestClassifier(n_estimators=100,random_state=0)
X=data[data.columns[:8]]
Y=data['Stress Level']
model.fit(X,Y)
pd.Series(model.feature_importances_,index=X.columns).sort_values(ascending=False)
```

Out[27]:
```
Stress Level    0.315796
Temperature     0.279227
Humidity        0.256394
Step count      0.148584
dtype: float64
```

In [28]:
```python
data2=data[['Stress Level','Temperature','Humidity','Step count']]
from sklearn.preprocessing import StandardScaler #Standardisation
features=data2[data2.columns[:4]]
features_standard=StandardScaler().fit_transform(features)# Gaussian Standardisation
x=pd.DataFrame(features_standard,columns=[['Stress Level','Temperature','Humidity','Step cou
nt']])
x['Stress Level']=data2['Stress Level']
outcome=x['Stress Level']
train1,test1=train_test_split(x,test_size=0.25,random_state=0,stratify=x['Stress Level'])
train_X1=train1[train1.columns[:4]]
test_X1=test1[test1.columns[:4]]
train_Y1=train1['Stress Level']
test_Y1=test1['Stress Level']
```

In [29]:
```python
abc=[]
classifiers=['Linear Svm','Radial Svm','Logistic Regression','KNN','Decision Tree']
models=[svm.SVC(kernel='linear'),svm.SVC(kernel='rbf'),LogisticRegression(),KNeighborsClassi
fier(n_neighbors=3),DecisionTreeClassifier()]
for i in models:
    model = i
    model.fit(train_X1,train_Y1)
    prediction=model.predict(test_X1)
    abc.append(metrics.accuracy_score(prediction,test_Y1))
new_models_dataframe=pd.DataFrame(abc,index=classifiers)
new_models_dataframe.columns=['New Accuracy']
```

In [30]:
```python
new_models_dataframe=new_models_dataframe.merge(models_dataframe,left_index=True,right_index
=True,how='left')
new_models_dataframe['Increase']=new_models_dataframe['New Accuracy']-new_models_dataframe[
'Accuracy']
new_models_dataframe
```

Out[30]:

	New Accuracy	Accuracy	Increase
Linear Svm	1.0	1.000000	0.000000
Radial Svm	1.0	1.000000	0.000000
Logistic Regression	1.0	1.000000	0.000000
KNN	1.0	0.998004	0.001996
Decision Tree	1.0	1.000000	0.000000

In [31]:
```python
#scale dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
train_X1= sc.fit_transform(train_X1)
test_X1=sc.transform(test_X1)
```

In [32]:
```python
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(C= 1.0, penalty='l2',random_state=0)
clf.fit(train_X1, train_Y1)
```

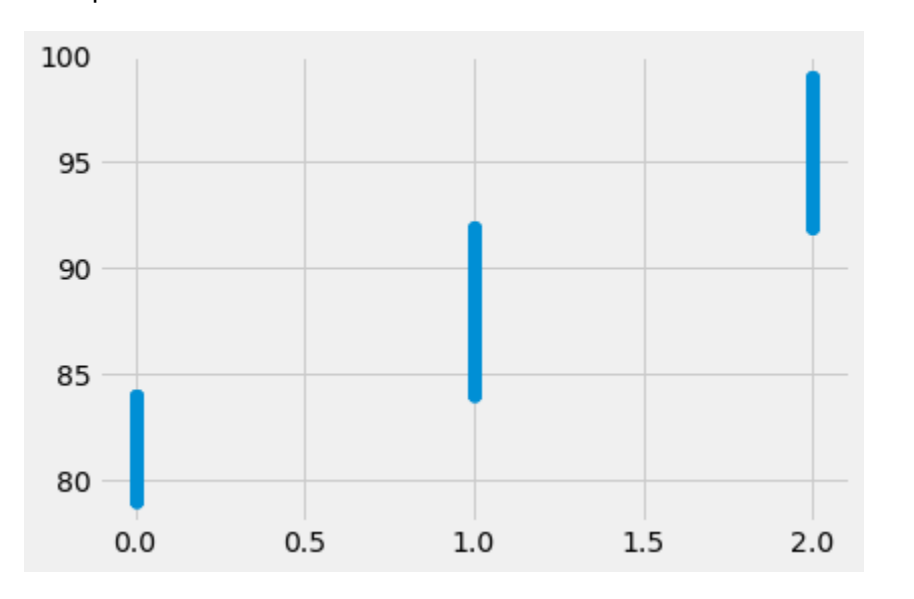
Out[32]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [33]:
```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
pred = clf.predict(test_X1)
cm = confusion_matrix(test_Y1,pred )
print(cm)
print(accuracy_score(test_Y1,pred))
```

```
[[125   0   0]
 [  0 198   0]
 [  0   0 178]]
1.0
```

In [34]:
```python
plt.scatter(data["Stress Level"], data["Temperature"])
```

Out[34]:
```
<matplotlib.collections.PathCollection at 0x9c8a410>
```



In [35]:
```python
X1 = data[data["Stress Level"] == 0]["Step count"]
X2 = data[data["Stress Level"] == 0]["Temperature"]
plt.scatter(X1, X2)
print(min(X1), max(X1))
```

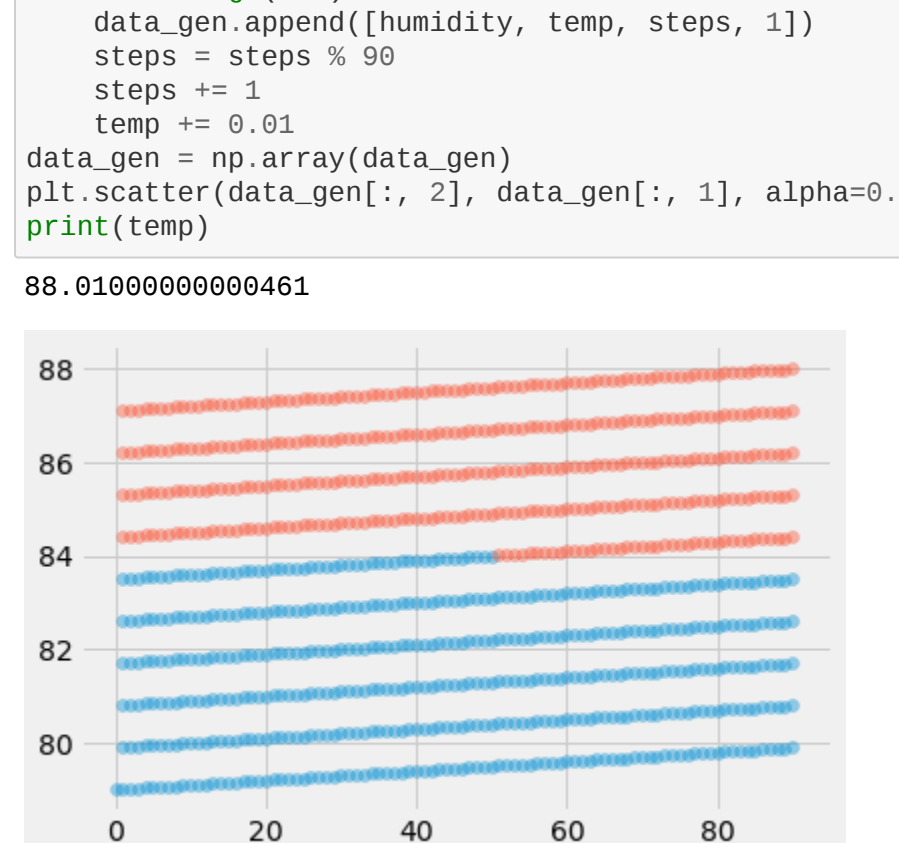
```
0 90
```

In [36]:
```python
data_gen = []
humidity, temp, steps = 10., 79., 0.
for i in range(501):
    data_gen.append([humidity, temp, steps, 0])
    steps = steps % 90
    steps += 1
    temp += 0.01

data_gen = np.array(data_gen)
plt.scatter(data_gen[:, 2], data_gen[:, 1], alpha=0.4)
data_gen = []

for i in range(400):
    data_gen.append([humidity, temp, steps, 1])
    steps = steps % 90
    steps += 1
    temp += 0.01
data_gen = np.array(data_gen)
plt.scatter(data_gen[:, 2], data_gen[:, 1], alpha=0.4)
print(temp)
```

```
88.01000000000461
```



In [37]:
```python
from sklearn.model_selection import KFold #for K-fold cross validation
from sklearn.model_selection import cross_val_score #score evaluation
```

In [38]:
```python
kfold = KFold(n_splits=10, random_state=None) # k=10, split the data into 10 equal parts
```

In [41]:
```python
xyz=[]
accuracy=[]
classifiers=['Linear Svm','Radial Svm','Logistic Regression','KNN','Decision Tree']
models=[svm.SVC(kernel='linear'),svm.SVC(kernel='rbf'),LogisticRegression(),KNeighborsClassi
fier(n_neighbors=3),DecisionTreeClassifier()]
for i in models:
    model = i
    cv_result = cross_val_score(model,x[x.columns[:4]],x['Stress Level'], cv = kfold,scoring
= "accuracy")
    cv_result=cv_result
    xyz.append(cv_result.mean())
    accuracy.append(cv_result)
new_models_dataframe2=pd.DataFrame(abc,index=classifiers)
new_models_dataframe2.columns=['CV Mean']
new_models_dataframe2
```

Out[41]:

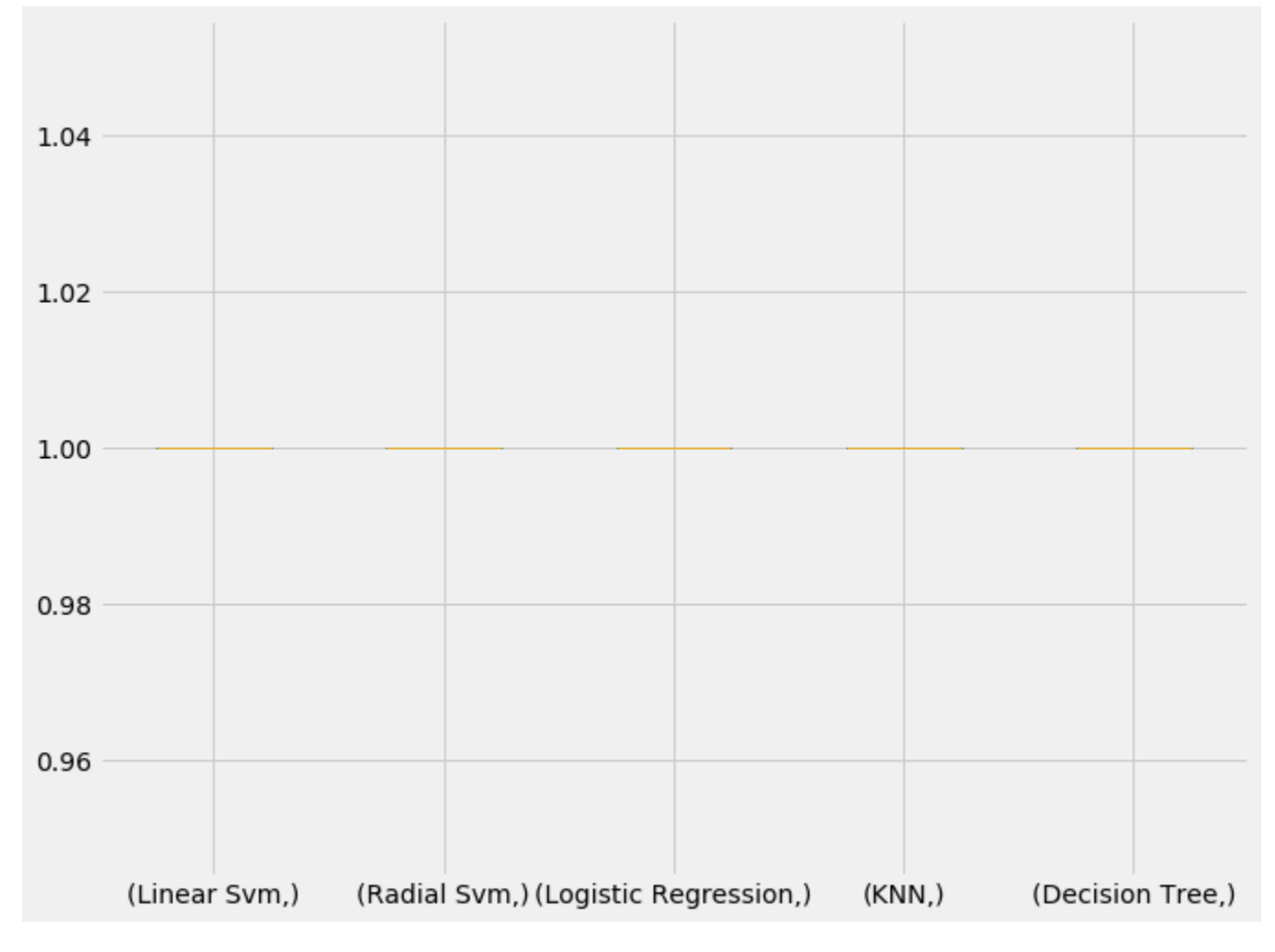
	CV Mean
Linear Svm	1.0
Radial Svm	1.0
Logistic Regression	1.0
KNN	1.0
Decision Tree	1.0

In [42]:
```python
box=pd.DataFrame(accuracy,index=[classifiers])
Box1=box.T
Box1
```

Out[42]:

	Linear Svm	Radial Svm	Logistic Regression	KNN	Decision Tree
0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0

In [43]:
```python
Box1.boxplot()
fig=plt.gcf()
fig.set_size_inches(10,8)
plt.show()
```



In [44]:
```python
linear_svc=svm.SVC(kernel='linear',C=0.1,gamma=10,probability=True)
radial_svm=svm.SVC(kernel='rbf',C=0.1,gamma=10,probability=True)
lr=LogisticRegression(C=0.1)
```

In [45]:
```python
from sklearn.ensemble import VotingClassifier #for Voting Classifier
```

In [46]:
```python
# Linear and Radial SVM
ensemble_lin_rbf=VotingClassifier(estimators=[('Linear_svm', linear_svc), ('Radial_svm', rad
ial_svm)],
                       voting='soft', weights=[2,1]).fit(train_X1,train_Y1)
print('The accuracy for Linear and Radial SVM is:',ensemble_lin_rbf.score(test_X1,test_Y1))
```

```
The accuracy for Linear and Radial SVM is: 1.0
```

In [47]:
```python
# Linear SVM with Logistic Regression
ensemble_lin_lr=VotingClassifier(estimators=[('Linear_svm', linear_svc), ('Logistic Regressi
on', lr)],
                       voting='soft', weights=[2,1]).fit(train_X1,train_Y1)
print('The accuracy for Linear SVM and Logistic Regression is:',ensemble_lin_lr.score(test_X
1,test_Y1))
```

```
The accuracy for Linear SVM and Logistic Regression is: 1.0
```

In [48]:
```python
# Logistic Regression with Radial SVM
ensemble_rad_lr=VotingClassifier(estimators=[('Radial_svm', radial_svm), ('Logistic Regressi
on', lr)],
                       voting='soft', weights=[1,2]).fit(train_X1,train_Y1)
print('The accuracy for Radial SVM and Logistic Regression is:',ensemble_rad_lr.score(test_X
1,test_Y1))
```

```
The accuracy for Radial SVM and Logistic Regression is: 1.0
```

In [49]:
```python
ensemble_rad_lr_lin=VotingClassifier(estimators=[('Radial_svm', radial_svm), ('Logistic Regr
ession', lr),('Linear_svm',linear_svc)],
                       voting='soft', weights=[2,1,3]).fit(train_X1,train_Y1)
print('The ensembled model with all the 3 classifiers is:',ensemble_rad_lr_lin.score(test_X1
,test_Y1))
```

```
The ensembled model with all the 3 classifiers is: 1.0
```

In [ ]: