In [1]:
```
# Author : Amir Shokri
# github link : https://github.com/amirshnll/COVID-19-Surveillance
# dataset link : http://archive.ics.uci.edu/ml/datasets/COVID-19+Surveillance
# email : amirsh.nll@gmail.com
```

Compare different models for predicting whether a couple will get divorced

-Decision Tree
-Logistic Regression
-Naive Bayes
-KNN
-MLP

The Dataset

The Dataset is from UCIMachinelearning and it provides you all the relevant information needed for the prediction of Divorce. It contains 54 features and on the basis of these features we have to predict that the couple has been divorced or not. Value 1 represent Divorced and value 0 represent not divorced. Features are as follows:

1. **If one of us apologizes when our discussion deteriorates, the discussion ends.**
2. **I know we can ignore our differences, even if things get hard sometimes.**
3. **When we need it, we can take our discussions with my spouse from the beginning and correct it.**
4. **When I discuss with my spouse, to contact him will eventually work.**
5. **The time I spent with my wife is special for us.**
6. **We don't have time at home as partners.**
7. **We are like two strangers who share the same environment at home rather than family.**
8. **I enjoy our holidays with my wife.**
9. **I enjoy traveling with my wife.**
10. **Most of our goals are common to my spouse.**
11. **I think that one day in the future, when I look back, I see that my spouse and I have been in harmony with each other.**
12. **My spouse and I have similar values in terms of personal freedom.**
13. **My spouse and I have similar sense of entertainment.**
14. **Most of our goals for people (children, friends, etc.) are the same.**
15. **Our dreams with my spouse are similar and harmonious.**
16. **We're compatible with my spouse about what love should be.**
17. **We share the same views about being happy in our life with my spouse**
18. **My spouse and I have similar ideas about how marriage should be**
19. **My spouse and I have similar ideas about how roles should be in marriage**
20. **My spouse and I have similar values in trust.**
21. **I know exactly what my wife likes.**
22. **I know how my spouse wants to be taken care of when she/he sick.**
23. **I know my spouse's favorite food.**
24. **I can tell you what kind of stress my spouse is facing in her/his life.**
25. **I have knowledge of my spouse's inner world.**
26. **I know my spouse's basic anxieties.**
27. **I know what my spouse's current sources of stress are.**
28. **I know my spouse's hopes and wishes.**
29. **I know my spouse very well.**
30. **I know my spouse's friends and their social relationships.**
31. **I feel aggressive when I argue with my spouse.**
32. **When discussing with my spouse, I usually use expressions such as 'you always' or 'you never' .**
33. **I can use negative statements about my spouse's personality during our discussions.**
34. **I can use offensive expressions during our discussions.**
35. **I can insult my spouse during our discussions.**
36. **I can be humiliating when we discussions.**
37. **My discussion with my spouse is not calm.**
38. **I hate my spouse's way of open a subject.**
39. **Our discussions often occur suddenly.**

40. We're just starting a discussion before I know what's going on.
41. When I talk to my spouse about something, my calm suddenly breaks.
42. When I argue with my spouse, ı only go out and I don't say a word.
43. I mostly stay silent to calm the environment a little bit.
44. Sometimes I think it's good for me to leave home for a while.
45. I'd rather stay silent than discuss with my spouse.
46. Even if I'm right in the discussion, I stay silent to hurt my spouse.
47. When I discuss with my spouse, I stay silent because I am afraid of not being able to control my anger.
48. I feel right in our discussions.
49. I have nothing to do with what I've been accused of.
50. I'm not actually the one who's guilty about what I'm accused of.
51. I'm not the one who's wrong about problems at home.
52. I wouldn't hesitate to tell my spouse about her/his inadequacy.
53. When I discuss, I remind my spouse of her/his inadequacy.
54. I'm not afraid to tell my spouse about her/his incompetence.

Generally, logistic Machine Learning in Python has a straightforward and user-friendly implementation. It usually consists of these steps:

1. Import packages, functions, and classes
2. Get data to work with and, if appropriate, transform it
3. Create a classification model and train (or fit) it with existing data
4. Evaluate your model to see if its performance is satisfactory
5. Apply your model to make predictions

Import packages, functions, and classes

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix as cm
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn import tree
```

Get data to work with and, if appropriate, transform it

In [2]:
```
df = pd.read_csv('divorce.csv',sep=';')
df.head()
```

Out[2]:

	Atr1	Atr2	Atr3	Atr4	Atr5	Atr6	Atr7	Atr8	Atr9	Atr10	...	Atr46	Atr47	Atr48	Atr49	A
0	2	2	4	1	0	0	0	0	0	0	...	2	1	3	3	
1	4	4	4	4	4	0	0	4	4	4	...	2	2	3	4	
2	2	2	2	2	1	3	2	1	1	2	...	3	2	3	1	
3	3	2	3	2	3	3	3	3	3	3	...	2	2	3	3	
4	2	2	1	1	1	1	0	0	0	0	...	2	1	2	3	

5 rows × 55 columns

In [3]:

```
df.info()
```

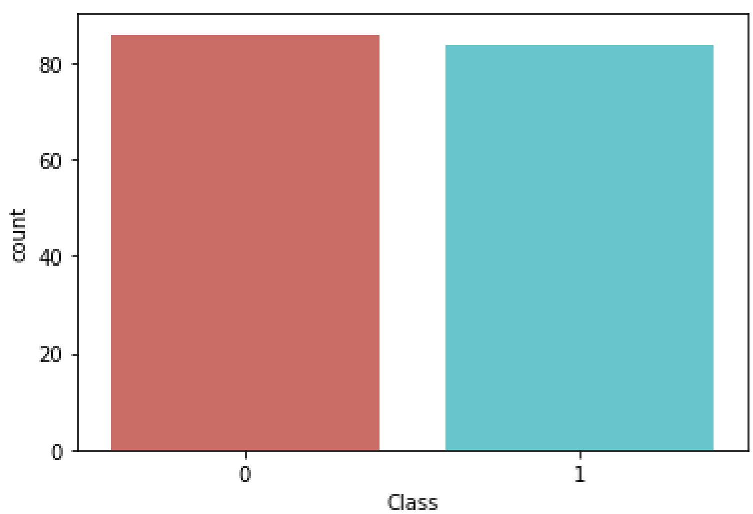
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170 entries, 0 to 169
Data columns (total 55 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Atr1    170 non-null    int64
 1   Atr2    170 non-null    int64
 2   Atr3    170 non-null    int64
 3   Atr4    170 non-null    int64
 4   Atr5    170 non-null    int64
 5   Atr6    170 non-null    int64
 6   Atr7    170 non-null    int64
 7   Atr8    170 non-null    int64
 8   Atr9    170 non-null    int64
 9   Atr10   170 non-null    int64
 10  Atr11   170 non-null    int64
 11  Atr12   170 non-null    int64
 12  Atr13   170 non-null    int64
 13  Atr14   170 non-null    int64
 14  Atr15   170 non-null    int64
 15  Atr16   170 non-null    int64
 16  Atr17   170 non-null    int64
 17  Atr18   170 non-null    int64
 18  Atr19   170 non-null    int64
 19  Atr20   170 non-null    int64
 20  Atr21   170 non-null    int64
 21  Atr22   170 non-null    int64
 22  Atr23   170 non-null    int64
 23  Atr24   170 non-null    int64
 24  Atr25   170 non-null    int64
 25  Atr26   170 non-null    int64
 26  Atr27   170 non-null    int64
 27  Atr28   170 non-null    int64
 28  Atr29   170 non-null    int64
 29  Atr30   170 non-null    int64
 30  Atr31   170 non-null    int64
 31  Atr32   170 non-null    int64
 32  Atr33   170 non-null    int64
 33  Atr34   170 non-null    int64
 34  Atr35   170 non-null    int64
 35  Atr36   170 non-null    int64
 36  Atr37   170 non-null    int64
 37  Atr38   170 non-null    int64
 38  Atr39   170 non-null    int64
 39  Atr40   170 non-null    int64
 40  Atr41   170 non-null    int64
 41  Atr42   170 non-null    int64
 42  Atr43   170 non-null    int64
 43  Atr44   170 non-null    int64
 44  Atr45   170 non-null    int64
 45  Atr46   170 non-null    int64
 46  Atr47   170 non-null    int64
 47  Atr48   170 non-null    int64
 48  Atr49   170 non-null    int64
 49  Atr50   170 non-null    int64
 50  Atr51   170 non-null    int64
 51  Atr52   170 non-null    int64
```

```
 52  Atr53    170 non-null    int64
 53  Atr54    170 non-null    int64
 54  Class    170 non-null    int64
dtypes: int64(55)
memory usage: 73.2 KB
```

In [4]:
```
y=df.Class
x_data=df.drop(columns=['Class'])
# print(x_data)
```

Data description

In [5]:
```
sns.countplot(x='Class',data=df,palette='hls')
plt.show()

count_no_sub = len(df[df['Class']==0])
count_sub = len(df[df['Class']==1])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("percentage of no divorce is", pct_of_no_sub*100)
pct_of_sub = count_sub/(count_no_sub+count_sub)
print("percentage of divorce", pct_of_sub*100)
```




```
percentage of no divorce is 50.588235294117645
percentage of divorce 49.411764705882355
```

Normalize data

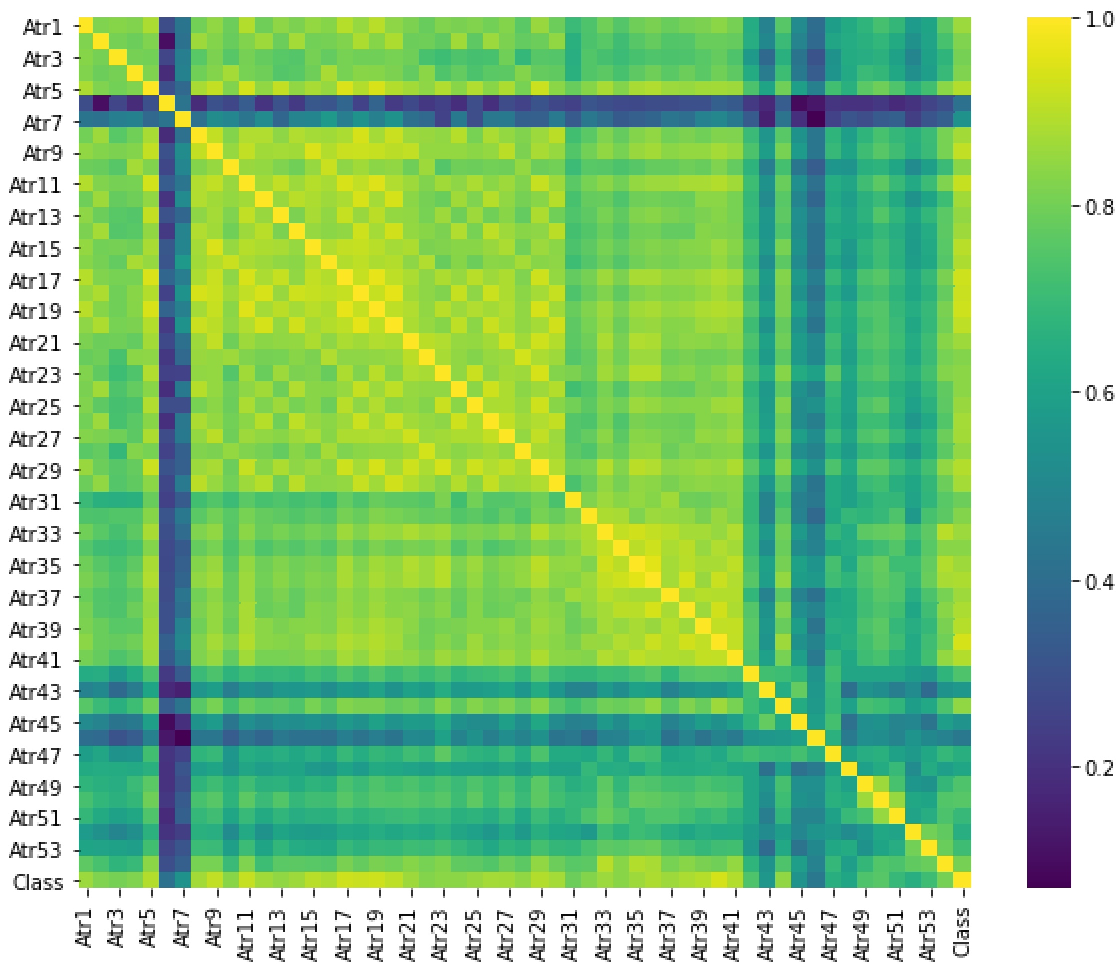
```python
In [6]: x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values
        x.head()
```

Out[6]:

	Atr1	Atr2	Atr3	Atr4	Atr5	Atr6	Atr7	Atr8	Atr9	Atr10	...	Atr45	Atr46	Atr47	Atr48	A
0	0.50	0.5	1.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	...	0.75	0.50	0.25	0.75	
1	1.00	1.0	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00	...	0.50	0.50	0.50	0.75	
2	0.50	0.5	0.50	0.50	0.25	0.75	0.50	0.25	0.25	0.50	...	0.50	0.75	0.50	0.75	
3	0.75	0.5	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.75	...	0.75	0.50	0.50	0.75	
4	0.50	0.5	0.25	0.25	0.25	0.25	0.00	0.00	0.00	0.00	...	0.50	0.50	0.25	0.50	

5 rows × 54 columns

```python
In [7]: plt.figure(figsize=(10,8))
        sns.heatmap(df.corr(), cmap='viridis');
```



Split dataset to data train & data test

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.4,random
_state=400)
print("x_train: ",x_train.shape)
print("x_test: ",x_test.shape)
print("y_train: ",y_train.shape)
print("y_test: ",y_test.shape)
```

```
x_train:  (102, 54)
x_test:  (68, 54)
y_train:  (102,)
y_test:  (68,)
```

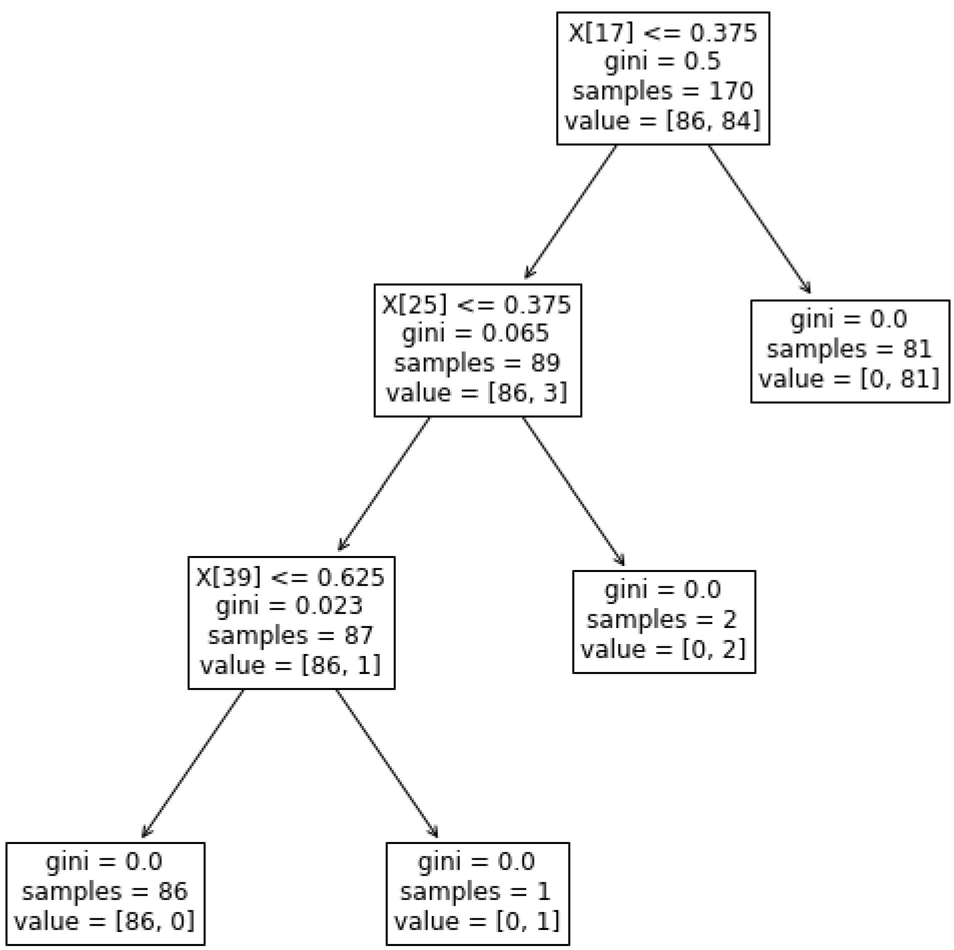
Train & Score

Step 1. Import the model you want to use
Step 2. Make an instance of the Model
Step 3. Training the model on the data, storing the information learned from the data
Step 4. Predict labels for new data

Decision Tree Classifier

In [9]:
```
clft = DecisionTreeClassifier()
clft = clft.fit(x_train,y_train)
y_predt = clft.predict(x_test)# step 4
print(classification_report(y_test, clft.predict(x_test)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'.format(clft.s
core(x_test, y_test)))
from sklearn import tree
plt.figure(figsize=(10,10))
temp = tree.plot_tree(clft.fit(x,y), fontsize=12)
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.97      0.94      0.95        33
           1       0.94      0.97      0.96        35

    accuracy                           0.96        68
   macro avg       0.96      0.96      0.96        68
weighted avg       0.96      0.96      0.96        68

Accuracy of Decision Tree classifier on test set: 0.96
```



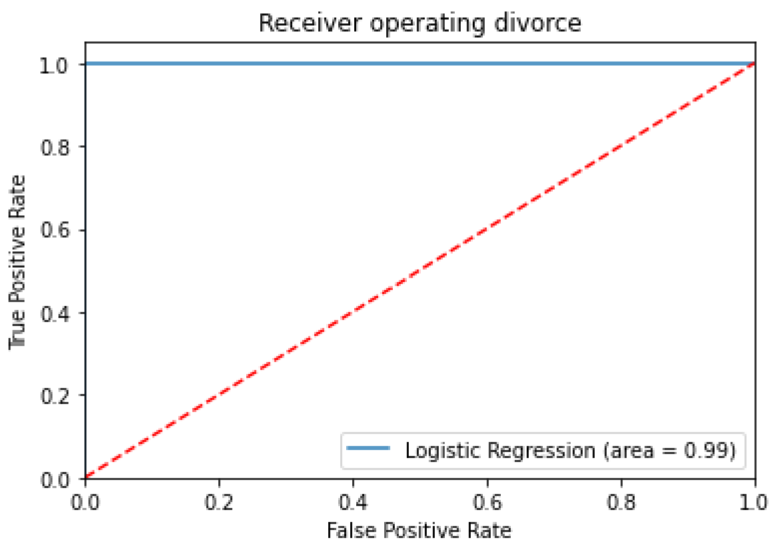
Logistic Regreession Classifier

In [10]:
```
clfr = LogisticRegression(solver='lbfgs')# step 2
clfr.fit(x_train, y_train.ravel())# step 3
y_predr = clfr.predict(x_test)# step 4
# model = LogisticRegression(solver='liblinear', random_state=0).fit(x_train,
 y_train.ravel())
```

In [11]:
```
print(classification_report(y_test, clfr.predict(x_test)))
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(
clfr.score(x_test, y_test)))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        33
           1       1.00      0.97      0.99        35

    accuracy                           0.99        68
   macro avg       0.99      0.99      0.99        68
weighted avg       0.99      0.99      0.99        68

Accuracy of logistic regression classifier on test set: 0.99
```

In [12]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, clfr.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test, clfr.predict_proba(x_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating divorce')
plt.legend(loc="lower right")
plt.show()
```



Naive Bayes Classifier

In [13]:
```
clfb = GaussianNB()
clfb.fit(x_train, y_train.ravel())
y_predb = clfb.predict(x_test)# step 4
print(classification_report(y_test, clfb.predict(x_test)))
print("Naive Bayes test accuracy: ", clfb.score(x_test, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       1.00      1.00      1.00        35

    accuracy                           1.00        68
   macro avg       1.00      1.00      1.00        68
weighted avg       1.00      1.00      1.00        68

Naive Bayes test accuracy:  1.0
```

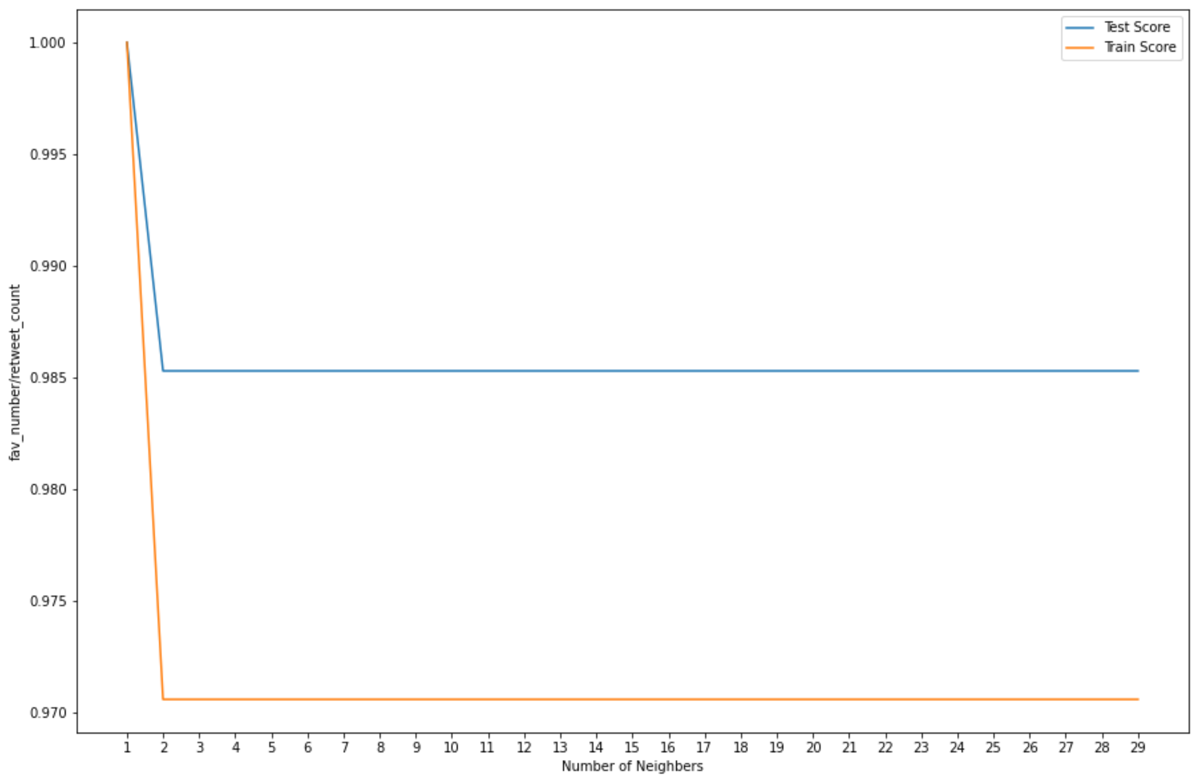
KNN Classifier

In [14]:
```
K = 5
clfk = KNeighborsClassifier(n_neighbors=K)
clfk.fit(x_train, y_train.ravel())
y_predk=clfk.predict(x_test)

print("When K = {} neighnors , KNN test accuracy: {}".format(K, clfk.score(x_t
est, y_test)))
print("When K = {} neighnors , KNN train accuracy: {}".format(K, clfk.score(x_
train, y_train)))
print(classification_report(y_test, clfk.predict(x_test)))
print("Knn(k=5) test accuracy: ", clfk.score(x_test, y_test))

ran = np.arange(1,30)
train_list = []
test_list = []
for i,each in enumerate(ran):
    clfk = KNeighborsClassifier(n_neighbors=each)
    clfk.fit(x_train, y_train.ravel())
    test_list.append(clfk.score(x_test, y_test))
    train_list.append(clfk.score(x_train, y_train))


print("Best test score is {} , K = {}".format(np.max(test_list), test_list.ind
ex(np.max(test_list))+1))
print("Best train score is {} , K = {}".format(np.max(train_list), train_list.
index(np.max(train_list))+1))
```

```
When K = 5 neighnors , KNN test accuracy: 0.9852941176470589
When K = 5 neighnors , KNN train accuracy: 0.9705882352941176
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        33
           1       1.00      0.97      0.99        35

    accuracy                           0.99        68
   macro avg       0.99      0.99      0.99        68
weighted avg       0.99      0.99      0.99        68

Knn(k=5) test accuracy:  0.9852941176470589
Best test score is 1.0 , K = 1
Best train score is 1.0 , K = 1
```

In [15]:
```
plt.figure(figsize=[15,10])
plt.plot(ran,test_list,label='Test Score')
plt.plot(ran,train_list,label = 'Train Score')
plt.xlabel('Number of Neighbers')
plt.ylabel('fav_number/retweet_count')
plt.xticks(ran)
plt.legend()
print("Best test score is {} , K = {}".format(np.max(test_list), test_list.ind
ex(np.max(test_list))+1))
print("Best train score is {} , K = {}".format(np.max(train_list), train_list.
index(np.max(train_list))+1))
```

```
Best test score is 1.0 , K = 1
Best train score is 1.0 , K = 1
```



MLP Classifier

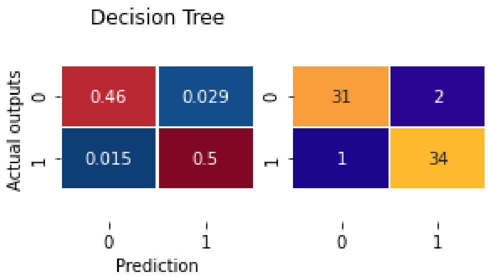
In [16]:
```
clfm = MLPClassifier(hidden_layer_sizes=(5,), max_iter=2000)
clfm.fit(x_train, y_train.ravel())
y_predm = clfm.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_predm))
print(classification_report(y_test, clfm.predict(x_test)))
print("MLP test accuracy: ", clfm.score(x_test, y_test))
```

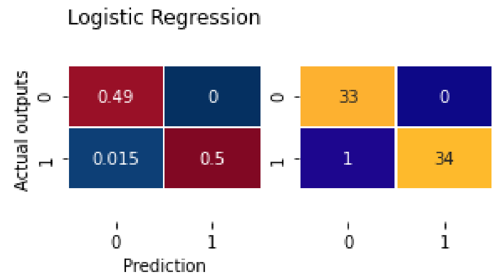
```
Accuracy: 0.9852941176470589
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        33
           1       1.00      0.97      0.99        35

    accuracy                           0.99        68
   macro avg       0.99      0.99      0.99        68
weighted avg       0.99      0.99      0.99        68

MLP test accuracy:  0.9852941176470589
```

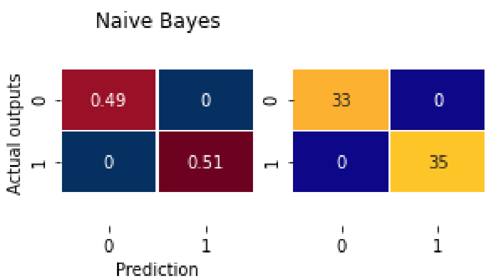
Compare Confusion Matrix

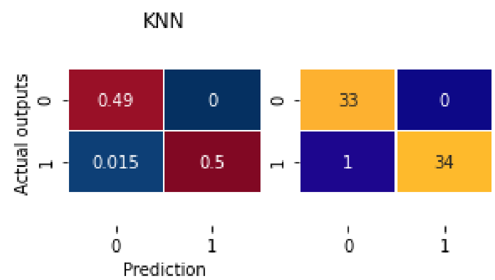
In [17]:
```
def confusionMatrix(y_pred,title,n):
    plt.subplot(5,5,n)
    ax=sns.heatmap(cm(y_test, y_pred)/sum(sum(cm(y_test, y_pred))), annot=True
                   ,cmap='RdBu_r', vmin=0, vmax=0.52,cbar=False, linewidths=.5
)
    plt.title(title)
    plt.ylabel('Actual outputs')
    plt.xlabel('Prediction')
    b, t=ax.get_ylim()
    ax.set_ylim(b+.5, t-.5)
    plt.subplot(5,5,n+1)
    axx=sns.heatmap(cm(y_test, y_pred), annot=True
                   ,cmap='plasma', vmin=0, vmax=40,cbar=False, linewidths=.5)
    b, t=axx.get_ylim()
    axx.set_ylim(b+.5, t-.5)
    return

plt.figure(figsize=(12,12))
# figure, axes = plt.subplots(nrows=1, ncols=1)
confusionMatrix(y_predt,'Decision Tree',1)
confusionMatrix(y_predr,'Logistic Regression',4)
confusionMatrix(y_predb,'Naive Bayes',11)
confusionMatrix(y_predk,'KNN',14)
confusionMatrix(y_predm,'MLP',21)
# plt.subplots_adjust(bottom=0.25, top=0.75)
# figure.tight_layout()
plt.savefig('Compare Confusion Matrix')
plt.show
```

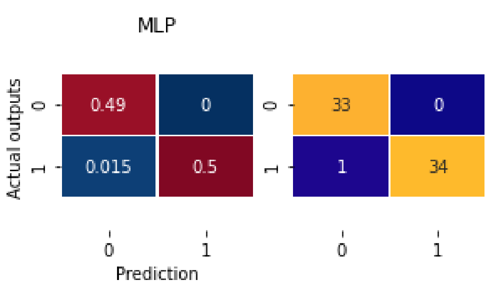
```
Out[17]: <function matplotlib.pyplot.show(close=None, block=None)>
```











Result:

So we have successfully trained our dataset into different models for predicting and compare whether a couple will get divorced or not in divorce data set. And also got the accuracy & confusion matrix for each model as well.