Shdg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step |  |  |  |  |
| 1 | Creating **predictors and target** sets | y = df['party'].values  X = df.drop('party', axis=1).values  y = y.reshape(-1, 1)  X = X.reshape(-1, 1) |  |  |
| 2 | Create **training and test** sets | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state=42) |  |  |
| 3 | Create the **regressor/classifier** (r/c) | knn = KNeighborsClassifier(n\_neighbors=6)  LinearRegression()  ridge = Ridge(alpha=0.1, normalize=True)  lasso = Lasso(alpha=0.1, normalize=True)  logreg = LogisticRegression()  tree = DecisionTreeClassifier()  elastic\_net = ElasticNet() |  |  |
| Cross validation for tuning | y\_pred\_prob = logreg.predict\_proba(X\_test)[:,1]  cv\_scores = cross\_val\_score(logreg, X, y, cv=5, scoring='roc\_auc') |  |  |
| GridSearchCV | In [2]: param\_grid = {'n\_neighbors': np.arange(1, 50)}  In [3]: knn = KNeighborsClassifier()  In [4]: knn\_cv = GridSearchCV(knn, param\_grid, cv=5) |  |  |
| 4 | **Fit** the r/c to the training data | knn.fit(X\_train, y\_train) # (Ridge/Lasso/LinearReg |  |  |
| **Cross validation** for fitting | cv\_results = cross\_val\_score(reg, X, y, cv=5)  np.mean(cv\_results)  In [5]: knn\_cv.fit(X, y)  In [6]: knn\_cv.best\_params\_ Out[6]: {'n\_neighbors': 12}  In [7]: knn\_cv.best\_score\_ Out[7]: 0.933216168717 |  |  |
| 5 | **Predict** on the test data | prediction = knn.predict(X\_test) |  |  |
| 6 | Compute the **accuracy** | knn.score(X\_test, y\_test)  Confusion Matrix:  confusion\_matrix(y\_test, y\_pred))  classification\_report(y\_test, y\_pred)  The ROC curve:  y\_pred\_prob = logreg.predict\_proba(X\_test)[:,1]  In [3]: fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_prob) |  |  |
| 7 |  |  |  |  |

|  |  |
| --- | --- |
| KNN | from sklearn.neighbors import KNeighborsClassifier |
| train\_test\_split | from sklearn.model\_selection import train\_test\_split |
| LinearRegression() | from sklearn import linear\_model |
| Mean squared error | from sklearn.metrics import mean\_squared\_error |
| Lasso | from sklearn.linear\_model import Lasso |
| Ridge | from sklearn.linear\_model import Ridge |
| cross\_val\_score | from sklearn.model\_selection import cross\_val\_score |
| LogisticRegression | from sklearn.linear\_model import LogisticRegression |
| GridSearchCV | from sklearn.model\_selection import GridSearchCV |
| ROC curve | from sklearn.metrics import roc\_auc\_score |
|  |  |
|  |  |