**Decide whether sport person will play or not on the basis of Outlook using KNN Algorithm**[**¶**](#gjdgxs)

In [1]:

**import** **pandas** **as** **pd**

**1. Acquire the data**[**¶**](#30j0zll)

In [2]:

df = pd.read\_csv('play.csv')  
df

Out[2]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **outlook** | **temperature** | **humidity** | **windy** | **play** |
| **0** | overcast | hot | high | False | yes |
| **1** | overcast | cool | normal | True | yes |
| **2** | overcast | mild | high | True | yes |
| **3** | overcast | hot | normal | False | yes |
| **4** | rainy | mild | high | False | yes |
| **5** | rainy | cool | normal | False | yes |
| **6** | rainy | cool | normal | True | no |
| **7** | rainy | mild | normal | False | yes |
| **8** | rainy | mild | high | True | no |
| **9** | sunny | hot | high | False | no |
| **10** | sunny | hot | high | True | no |
| **11** | sunny | mild | high | False | no |
| **12** | sunny | cool | normal | False | yes |
| **13** | sunny | mild | normal | True | yes |

In [3]:

X = df.drop('play', axis = 1)  
y = df['play']

In [4]:

X.head()

Out[4]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **outlook** | **temperature** | **humidity** | **windy** |
| **0** | overcast | hot | high | False |
| **1** | overcast | cool | normal | True |
| **2** | overcast | mild | high | True |
| **3** | overcast | hot | normal | False |
| **4** | rainy | mild | high | False |

In [5]:

y.head()

Out[5]:

0 yes  
1 yes  
2 yes  
3 yes  
4 yes  
Name: play, dtype: object

**2) Preprocess the data**[**¶**](#1fob9te)

In [6]:

**from** **sklearn.preprocessing** **import** LabelEncoder  
Encoder\_X = LabelEncoder()   
**for** col **in** X.columns:  
 X[col] = Encoder\_X.fit\_transform(X[col])  
X.head()

Out[6]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **outlook** | **temperature** | **humidity** | **windy** |
| **0** | 0 | 1 | 0 | 0 |
| **1** | 0 | 0 | 1 | 1 |
| **2** | 0 | 2 | 0 | 1 |
| **3** | 0 | 1 | 1 | 0 |
| **4** | 1 | 2 | 0 | 0 |

In [7]:

Encoder\_y=LabelEncoder()  
y = Encoder\_y.fit\_transform(y)

In [8]:

y

Out[8]:

array([1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1])

In [9]:

type(y)

Out[9]:

numpy.ndarray

In [10]:

type(X)

Out[10]:

pandas.core.frame.DataFrame

In [11]:

**from** **sklearn.model\_selection** **import** train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

In [12]:

df.describe()

Out[12]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **outlook** | **temperature** | **humidity** | **windy** | **play** |
| **count** | 14 | 14 | 14 | 14 | 14 |
| **unique** | 3 | 3 | 2 | 2 | 2 |
| **top** | rainy | mild | high | False | yes |
| **freq** | 5 | 6 | 7 | 8 | 9 |

**3) Train the model**[**¶**](#3znysh7)

In [13]:

**from** **sklearn.neighbors** **import** KNeighborsClassifier  
my\_model = KNeighborsClassifier(n\_neighbors=1)  
result = my\_model.fit(X\_train,y\_train)

**4) Test the model**[**¶**](#2et92p0)

In [14]:

predictions = result.predict(X\_test)

In [15]:

**from** **sklearn.metrics** **import** classification\_report,confusion\_matrix  
confusion\_matrix(y\_test,predictions)

Out[15]:

array([[1, 1],  
 [0, 1]], dtype=int64)

In [16]:

**from** **sklearn** **import** metrics  
print("Accuracy:",metrics.accuracy\_score(y\_test, predictions))

Accuracy: 0.6666666666666666

**5)Deploy the model**[**¶**](#tyjcwt)

In [17]:

pred\_new = result.predict([[0,1,1,0]])  
pred\_new

Out[17]:

array([1])

In [18]:

pred\_new = result.predict([[1,2,1,1]])  
pred\_new

Out[18]:

array([0])

In [19]:

pred\_new = result.predict([[1,2,2,1]])  
pred\_new

Out[19]:

array([1])

In [ ]:

In [ ]: