In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
df=pd.read_csv('C:\\ds project\\water_potability.csv')
df
```

Out[1]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_car
0	NaN	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379
1	3.716080	129.422921	18630.05786	6.635246	NaN	592.885359	15.180
2	8.099124	224.236259	19909.54173	9.275884	NaN	418.606213	16.868
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558
...	...	...	...	...	...	...	
3271	4.668102	193.681736	47580.99160	7.166639	359.948574	526.424171	13.894
3272	7.808856	193.553212	17329.80216	8.061362	NaN	392.449580	19.903
3273	9.419510	175.762646	33155.57822	7.350233	NaN	432.044783	11.039
3274	5.126763	230.603758	11983.86938	6.303357	NaN	402.883113	11.168
3275	7.874671	195.102299	17404.17706	7.509306	NaN	327.459761	16.140

3276 rows × 10 columns

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

In [4]:
```
df["ph"].fillna(df["ph"].mean(skipna=True), inplace=True)
df["Sulfate"].fillna(df["Sulfate"].mean(skipna=True), inplace=True)
df["Trihalomethanes"].fillna(df["Trihalomethanes"].mean(skipna=True), inpla
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

In [6]:
```python
from sklearn.model_selection import train_test_split
X = df.drop('Potability', axis=1)
y = df['Potability']
print(X)
print(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

```
            ph    Hardness        Solids  Chloramines     Sulfate  \
0     7.080795  204.890456  20791.31898     7.300212  368.516441
1     3.716080  129.422921  18630.05786     6.635246  333.775777
2     8.099124  224.236259  19909.54173     9.275884  333.775777
3     8.316766  214.373394  22018.41744     8.059332  356.886136
4     9.092223  181.101509  17978.98634     6.546600  310.135738
...        ...         ...          ...          ...         ...
3271  4.668102  193.681736  47580.99160     7.166639  359.948574
3272  7.808856  193.553212  17329.80216     8.061362  333.775777
3273  9.419510  175.762646  33155.57822     7.350233  333.775777
3274  5.126763  230.603758  11983.86938     6.303357  333.775777
3275  7.874671  195.102299  17404.17706     7.509306  333.775777

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity
0       564.308654       10.379783        86.990970   2.963135
1       592.885359       15.180013        56.329076   4.500656
2       418.606213       16.868637        66.420093   3.055934
3       363.266516       18.436525       100.341674   4.628771
4       398.410813       11.558279        31.997993   4.075075
...            ...             ...              ...        ...
3271    526.424171       13.894419        66.687695   4.435821
3272    392.449580       19.903225        66.396293   2.798243
3273    432.044783       11.039070        69.845400   3.298875
3274    402.883113       11.168946        77.488213   4.708658
3275    327.459761       16.140368        78.698446   2.309149

[3276 rows x 9 columns]
0       0
1       0
2       0
3       0
4       0
       ..
3271    1
3272    1
3273    1
3274    1
3275    1
Name: Potability, Length: 3276, dtype: int64
```

In [7]:
```python
# LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,y_train)
```

Out[7]:
```
LogisticRegression()
```

In [8]:
```
lr= model.score(X_test,y_test) * 100
print("Accuracy:",lr)
```

```
Accuracy: 62.80487804878049
```

In [9]:
```
#GAUSSIAN
from sklearn.naive_bayes import GaussianNB
classifire=GaussianNB()
classifire.fit(X_train,y_train)
```

Out[9]:
```
GaussianNB()
```

In [10]:
```
vr=classifire.score(X_test,y_test)*100
print("Accuracy:",vr)
```

```
Accuracy: 63.109756097560975
```

In [11]:
```
#DECISIONTREE CLASSIFIER
from sklearn.tree import DecisionTreeClassifier
modelDT = DecisionTreeClassifier()
modelDT.fit(X_train,y_train)
```

Out[11]:
```
DecisionTreeClassifier()
```

In [12]:
```
vr1=modelDT.score(X_test,y_test)*100
print("Accuracy:",vr1)
```

```
Accuracy: 56.40243902439024
```

In [13]:
```
#RANDOM FOREST CLASSIFIER
from sklearn.ensemble import RandomForestClassifier
modelRF = RandomForestClassifier()
modelRF.fit(X_train,y_train)
```

Out[13]:
```
RandomForestClassifier()
```

In [14]:
```
r1=modelRF.score(X_test,y_test)*100
print("Accuracy:",r1)
```

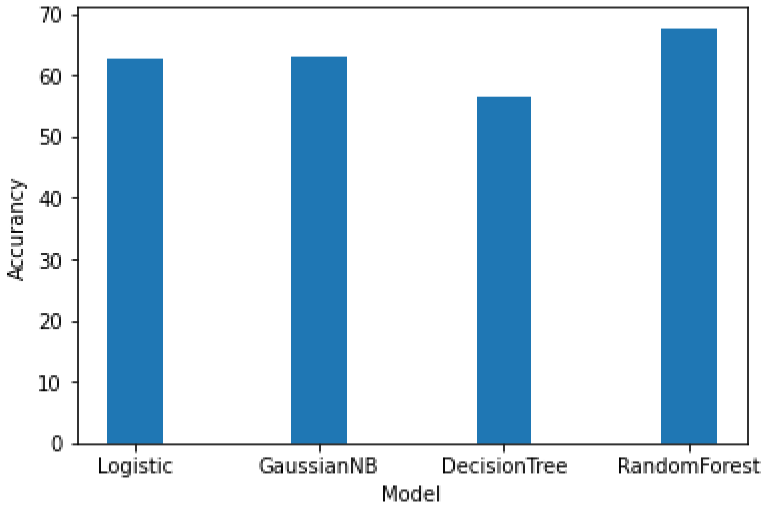
```
Accuracy: 67.6829268292683
```

In [15]:
```
print("hello")
```

```
hello
```

In [16]:

```
X=["Logistic","GaussianNB","DecisionTree","RandomForest"]
Y=[lr,vr,vr1,r1]
plt.bar(X,Y,width=0.3)
plt.xlabel("Model")
plt.ylabel("Accurancy")
plt.show()
```



In [19]:
```
ph = eval(input())
h = eval(input())
s= eval(input())
c = eval(input())
su = eval(input())
co = eval(input())
o = eval(input())
t = eval(input())
tu = eval(input())

a = {
    'ph': [ph],
    'Hardness': [h],
    'Solids': [s],
    'Chloramines': [c],
    'Sulfate': [su],
    'Conductivity': [co],
    'Organic_carbon': [o],
    'Trihalomethanes': [t],
    'Turbidity': [tu]
}

res = modelRF.predict(pd.DataFrame(a))

if res:
    print("POTABLE,YOU CAN!!!")
else:
    print("IMPOTABILITY,YOU MAY USE IT FOR GARDENING OR RECYCLE IT!!!")
```

```
23.4
34.5
65.7
65.43
32.11
32.23
34.321
234.56
56.78
POTABLE,YOU CAN!!!
```

In [ ]: