set up

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.under_sampling import RandomUnderSampler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, c
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Importing data

In [2]:
```
df = pd.read_csv('./data/magic.csv', header=None)
df = df.rename(columns={df.columns[-1]: 'class'})
df
```

Out[2]:

	0	1	2	3	4	5	6	7	8	
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4
...	...	...	...	...	...	...	...	...	...	
19015	21.3846	10.9170	2.6161	0.5857	0.3934	15.2618	11.5245	2.8766	2.4229	106.8
19016	28.9452	6.7020	2.2672	0.5351	0.2784	37.0816	13.1853	-2.9632	86.7975	247.4
19017	75.4455	47.5305	3.4483	0.1417	0.0549	-9.3561	41.0562	-9.4662	30.2987	256.5
19018	120.5135	76.9018	3.9939	0.0944	0.0683	5.8043	-93.5224	-63.8389	84.6874	408.3
19019	187.1814	53.0014	3.2093	0.2876	0.1539	-167.3125	-168.4558	31.4755	52.7310	272.3

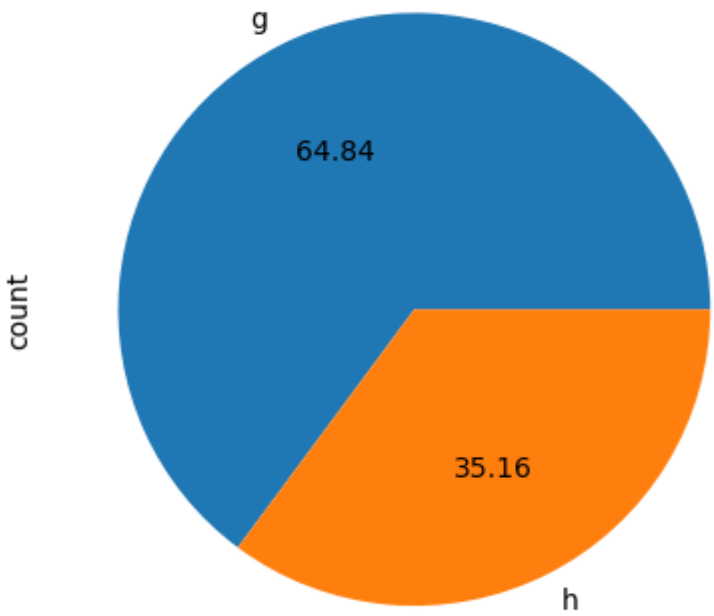
19020 rows × 11 columns

In [3]:
```
df['class'].value_counts()
```

Out[3]:
```
class
g    12332
h     6688
Name: count, dtype: int64
```

In [4]:
```
df['class'].value_counts().plot.pie(autopct='%.2f')
```

Out[4]:
```
<Axes: ylabel='count'>
```



what can we see from this plot ?

- we can see that the data is not balanced, around 64% of the data is classified as class 'g'
- this will add bias to the model and as a conclusion, the model accuracy will be bad.
- that's why we need to balance the data first to be able to continue our project

Encoding classes

In [5]:
```
df1 = df.copy()
df1['is_gamma'] = df['class'].apply(lambda x: 1 if x == 'g' else 0)
df1.drop(['class'], axis=1, inplace=True)
```

Balancing data

In [6]:
```
y = df1['is_gamma']
x = df1.drop(['is_gamma'], axis=1)
x
```

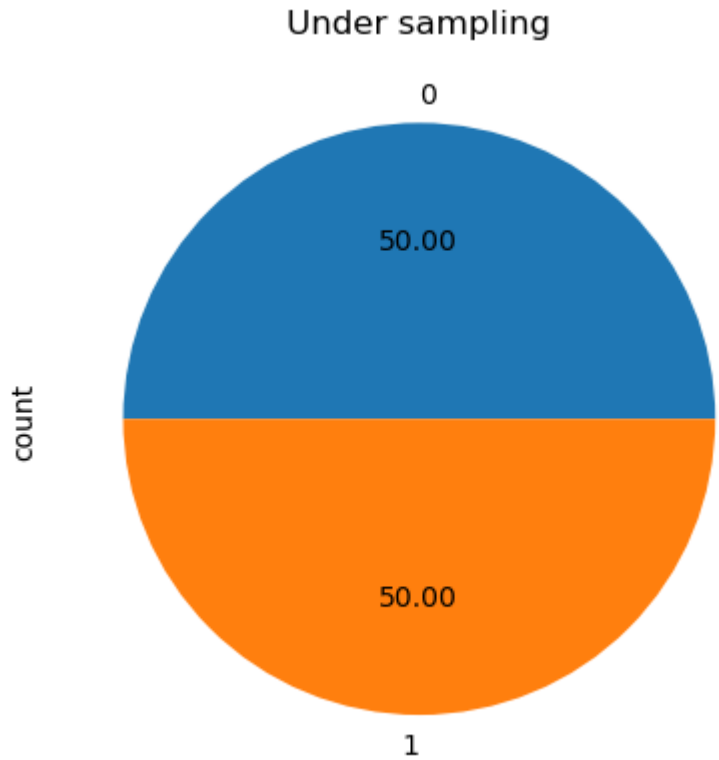
Out[6]:

	0	1	2	3	4	5	6	7	8	
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4
...	...	...	...	...	...	...	...	...	...	
19015	21.3846	10.9170	2.6161	0.5857	0.3934	15.2618	11.5245	2.8766	2.4229	106.8
19016	28.9452	6.7020	2.2672	0.5351	0.2784	37.0816	13.1853	-2.9632	86.7975	247.4
19017	75.4455	47.5305	3.4483	0.1417	0.0549	-9.3561	41.0562	-9.4662	30.2987	256.5
19018	120.5135	76.9018	3.9939	0.0944	0.0683	5.8043	-93.5224	-63.8389	84.6874	408.3
19019	187.1814	53.0014	3.2093	0.2876	0.1539	-167.3125	-168.4558	31.4755	52.7310	272.3

19020 rows × 10 columns

In [7]:
```
rus = RandomUnderSampler(sampling_strategy=1)
x_res, y_res = rus.fit_resample(x, y)

# plotting
ax = y_res.value_counts().plot.pie(autopct = '%.2f')
_=ax.set_title("Under sampling")
```



Data splitting

In [8]:
```
x_train, x_validationAndTesting, y_train, y_validationAndTesting = train_tes
x_validation,x_test, y_validation,  y_test = train_test_split(x_validationAr
```

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_validation = scaler.fit_transform(x_validation)
x_test = scaler.fit_transform(x_test)
```

KNN Classification

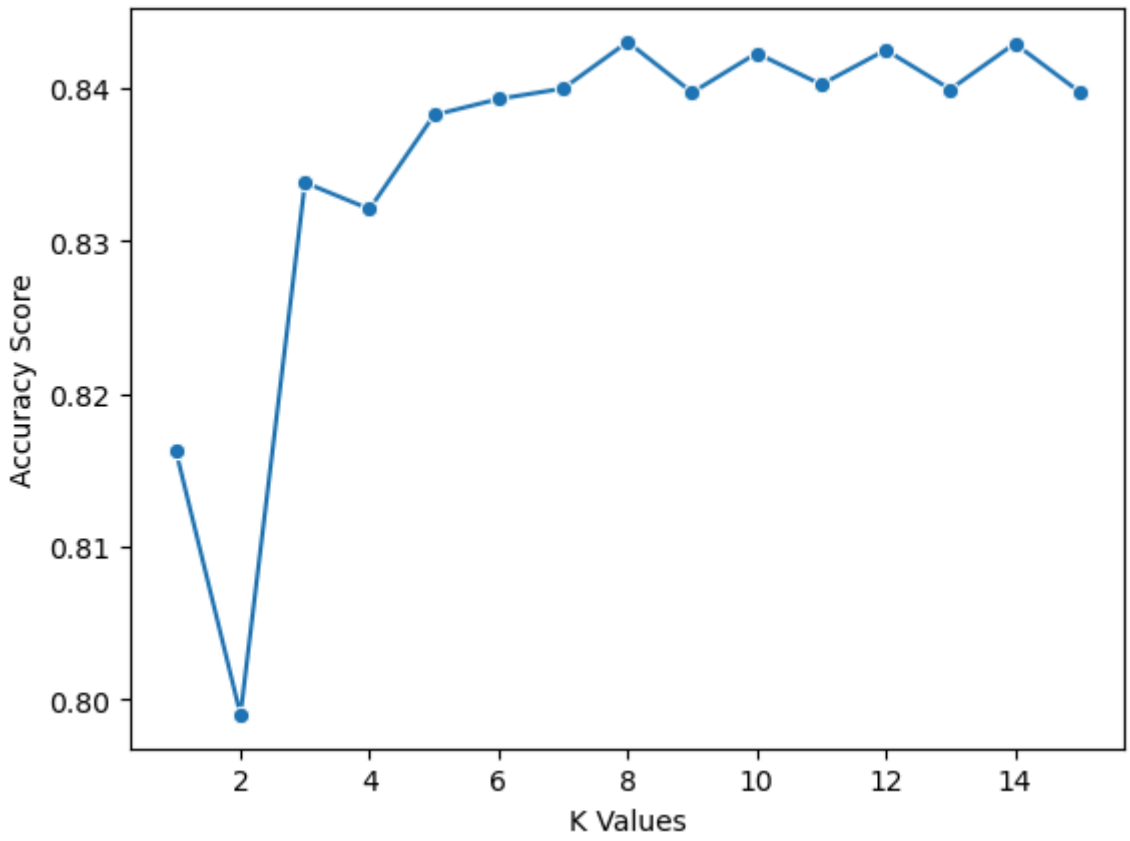
In [9]:
```
k_values = [i for i in range (1,16)]
scores = []

scaler = StandardScaler()
X = scaler.fit_transform(x)

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X, y, cv=5)
    scores.append(np.mean(score))

sns.lineplot(x = k_values, y = scores, marker = 'o')
plt.xlabel("K Values")
plt.ylabel("Accuracy Score")
```

Out[9]:
```
Text(0, 0.5, 'Accuracy Score')
```



In [10]:
```
best_index = np.argmax(scores)
best_k = k_values[best_index]

knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
print('best k value: ',best_k)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

```
best k value:  8
Accuracy: 0.804185351270553
Precision: 0.7723502304147466
Recall: 0.8516260162601627
```

Report

In [11]:
```
y_pred = knn.predict(x_test)

print('K value used: ', best_k)
print(classification_report(y_test, y_pred))
```

```
K value used:  8
              precision    recall  f1-score   support

           0       0.84      0.76      0.80      1023
           1       0.77      0.85      0.81       984

    accuracy                           0.80      2007
   macro avg       0.81      0.81      0.80      2007
weighted avg       0.81      0.80      0.80      2007
```