In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.spatial import Voronoi, voronoi_plot_2d
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [2]:

```
dataset=pd.read_csv("ParisHousingClass.csv")
```

In [3]:

```
dataset
```

Out[3]:

	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	cityPartRange	numPrevOwners	m
0	75523	3	0	1	63	9373	3	8	2
1	80771	39	1	1	98	39381	8	6	2
2	55712	58	0	1	19	34457	6	8	2
3	32316	47	0	0	6	27939	10	4	2
4	70429	19	1	1	90	38045	3	7	1
...	...	...	...	...	...	...	...	...	
9995	1726	89	0	1	5	73133	7	6	2
9996	44403	29	1	1	12	34606	9	4	1
9997	83841	3	0	0	69	80933	10	10	2
9998	59036	70	0	0	96	55856	1	3	2
9999	1440	84	0	0	49	18412	6	10	1

10000 rows × 18 columns

In [4]:

```
dataset.head()
```

Out[4]:

	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	cityPartRange	numPrevOwners	made
0	75523	3	0	1	63	9373	3	8	2005
1	80771	39	1	1	98	39381	8	6	2015
2	55712	58	0	1	19	34457	6	8	2021
3	32316	47	0	0	6	27939	10	4	2012
4	70429	19	1	1	90	38045	3	7	1990

In [5]:
```
dataset.describe()
```

Out[5]:

	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	cityPartRang
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	49870.13120	50.358400	0.508700	0.496800	50.276300	50225.486100	5.51010
std	28774.37535	28.816696	0.499949	0.500015	28.889171	29006.675799	2.87202
min	89.00000	1.000000	0.000000	0.000000	1.000000	3.000000	1.00000
25%	25098.50000	25.000000	0.000000	0.000000	25.000000	24693.750000	3.00000
50%	50105.50000	50.000000	1.000000	0.000000	50.000000	50693.000000	5.00000
75%	74609.75000	75.000000	1.000000	1.000000	76.000000	75683.250000	8.00000
max	99999.00000	100.000000	1.000000	1.000000	100.000000	99953.000000	10.00000

In [6]:
```
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

In [7]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

In [8]:
```
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

In [9]:
```
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In [11]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

Out[11]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [12]:
```
y_pred= classifier.predict(x_test)
```

In [13]:
```
y_pred
```

Out[13]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

```
In [14]: from sklearn.metrics import confusion_matrix
         cm= confusion_matrix(y_test, y_pred)
```

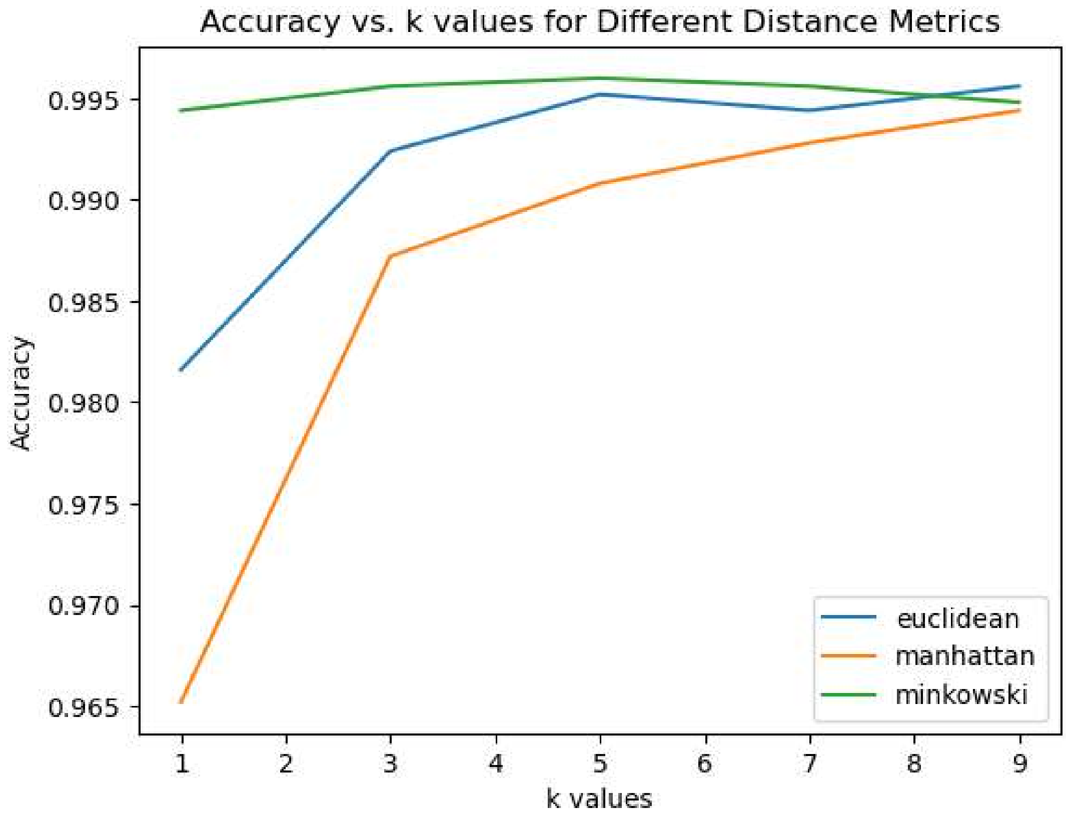
```
In [15]: cm
```

```
Out[15]: array([[2180,    0],
                [  12,  308]], dtype=int64)
```

```
In [16]: distance_metrics = ['euclidean', 'manhattan', 'minkowski']
```

```
In [17]: k_values = [1,3,5,7,9]
```

```
In [18]: for metric in distance_metrics:
             accuracy_values = []
             for k in k_values:
                 if metric == 'minkowski':
                     knn=KNeighborsClassifier (n_neighbors=k, metric=metric, p=3)
                 else:
                     knn = KNeighborsClassifier (n_neighbors=k, metric=metric)
                 knn.fit(x_train, y_train)
                 y_pred = knn.predict(x_test)
                 accuracy = metrics.accuracy_score (y_test, y_pred)
                 accuracy_values.append(accuracy)
             plt.plot(k_values, accuracy_values, label=metric)
         plt.xlabel('k values')
         plt.ylabel('Accuracy')
         plt.title('Accuracy vs. k values for Different Distance Metrics')
         plt.legend()
         plt.show()
```

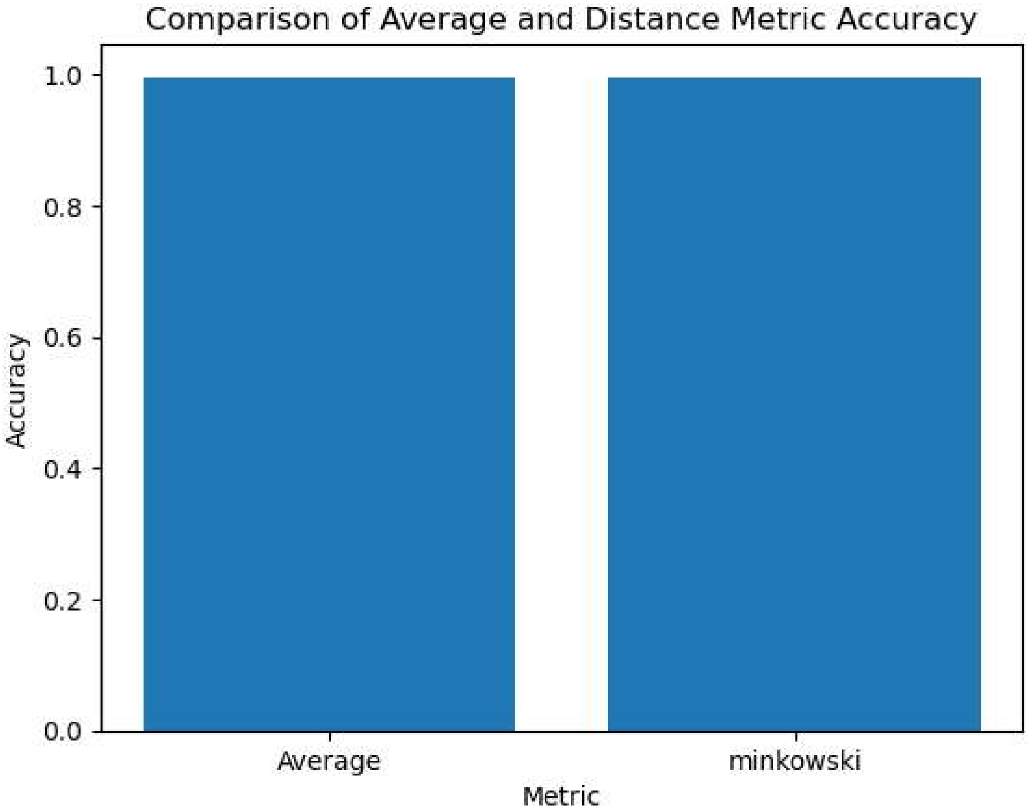


In [19]:

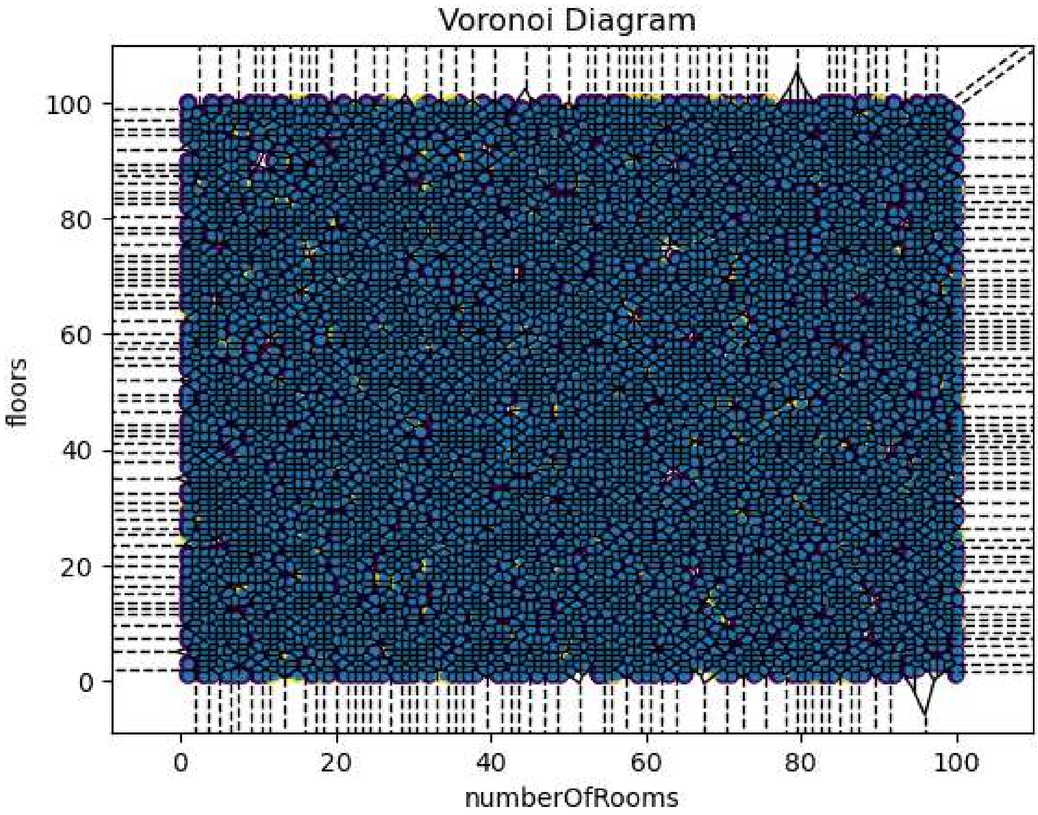
```
average_accuracy=np.mean(accuracy_values)
distance_metric_accuracy=accuracy_values[3]
```

In [20]:

```
plt.bar(['Average', distance_metrics [-1]], [average_accuracy, distance_metric_accuracy]
plt.xlabel('Metric')
plt.ylabel('Accuracy')
plt.title('Comparison of Average and Distance Metric Accuracy')
plt.show()
```



In [31]:
```
points = dataset[['numberOfRooms', 'floors']].values
vor = Voronoi(points)
voronoi_plot_2d (vor, show_vertices =False, show_points=True)
plt.scatter(dataset['numberOfRooms'], dataset['floors'], c=y, cmap='viridis')
plt.title('Voronoi Diagram ')
plt.xlabel('numberOfRooms')
plt.ylabel('floors')
plt.show()
```



In [ ]: