In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

from scipy.stats import mode
import warnings
```

In [3]:
```
data=pd.read_csv('iphone.csv')
data.head(10)
```

Out[3]:

	Gender	Age	Salary	Purchase Iphone
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
5	Male	27	58000	0
6	Female	27	84000	0
7	Female	32	150000	1
8	Male	25	33000	0
9	Female	35	65000	0

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Gender           400 non-null    object
 1   Age              400 non-null    int64 
 2   Salary           400 non-null    int64 
 3   Purchase Iphone  400 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

In [5]:
```
data.shape
```

Out[5]: (400, 4)

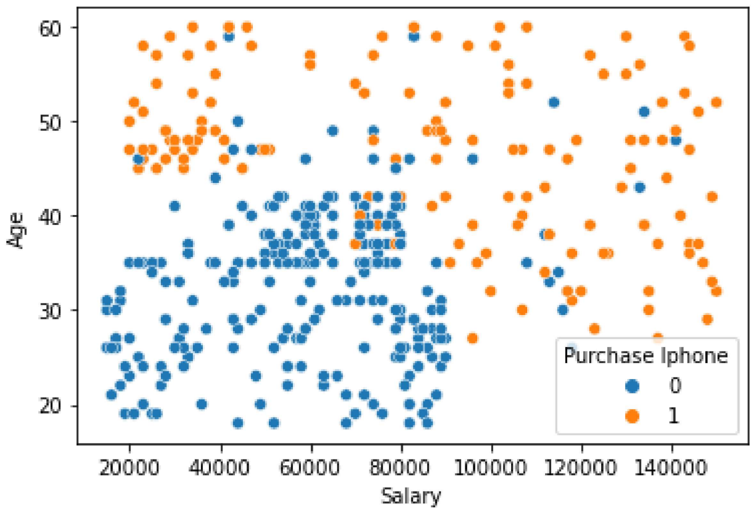
In [6]:
```
data.describe()
```

Out[6]:

	Age	Salary	Purchase Iphone
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

In [7]:
```
sns.scatterplot(x=data['Salary'], y=data['Age'], hue=data['Purchase Iphone'])
```

Out[7]: <AxesSubplot:xlabel='Salary', ylabel='Age'>



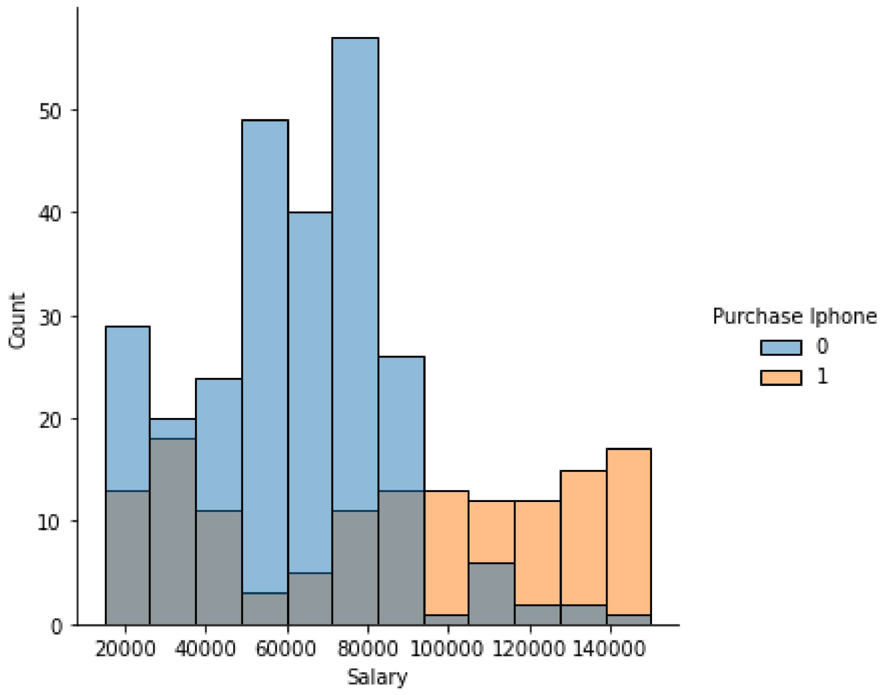
In [8]:
```
data = data.drop('Gender',axis=1)
data.head()
```

Out[8]:

	Age	Salary	Purchase Iphone
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

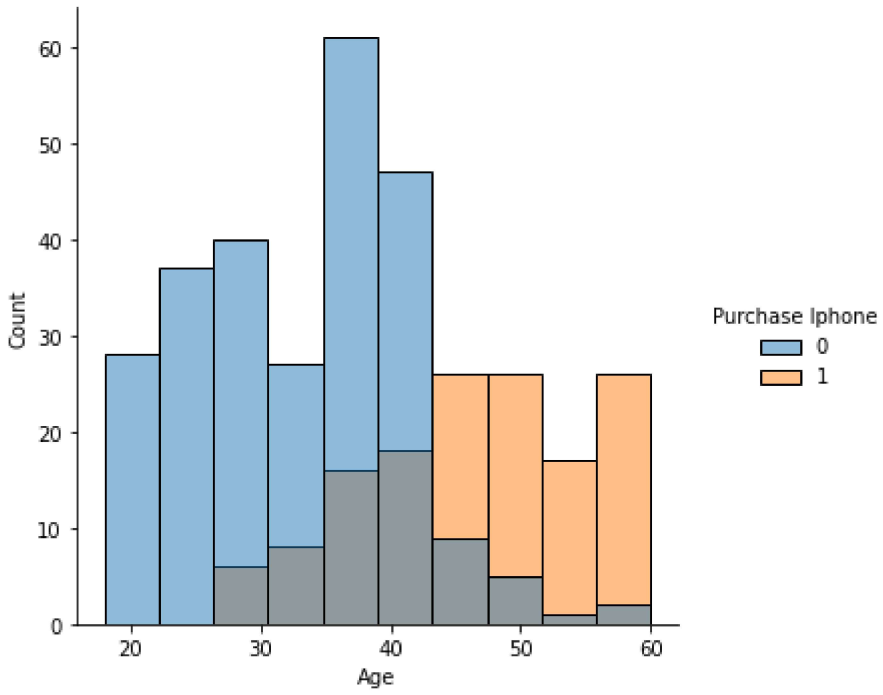
In [9]:
```
sns.displot(data, x='Salary', hue='Purchase Iphone')
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x152d0f1f220>



In [10]:
```
sns.displot(data,x='Age', hue='Purchase Iphone')
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x152d2fd2eb0>



In [11]:
```
import numpy as np
def euclidean_distance(pt1, pt2):
    distance = np.sqrt(np.sum((pt1 - pt2) ** 2))
    return distance

a=np.array([3, 5])
b=np.array([5, 9])

print(euclidean_distance(a, b))
```

```
4.47213595499958
```

In [12]:
```
X = data.drop('Purchase Iphone', axis=1)
Y = data['Purchase Iphone']
```

In [13]:
```
print(X)
```

```
     Age  Salary
0     19   19000
1     35   20000
2     26   43000
3     27   57000
4     19   76000
..   ...     ...
395   46   41000
396   51   23000
397   50   20000
398   36   33000
399   49   36000

[400 rows x 2 columns]
```

In [14]:
```
print(Y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchase Iphone, Length: 400, dtype: int64
```

In [15]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.35,random_state=0)
```

In [16]:
```
print(X_train.shape)
```

```
(260, 2)
```

In [17]:
```
print(y_train.shape)
```

```
(260,)
```

In [18]:
```
print(X_test.shape)
```

```
(140, 2)
```

In [19]:
```
print(y_test.shape)
```

```
(140,)
```

In [20]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [21]:
```
unknown_value = KNeighborsClassifier(n_neighbors=5)
```

In [22]:
```
unknown_value.fit(X_train, y_train)
```

Out[22]:
```
KNeighborsClassifier()
```

In [23]:
```
y_pred = unknown_value.predict(X_test)
y_pred
```

Out[23]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1], dtype=int64)
```

In [24]:
```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:',accuracy)
```

```
Accuracy: 0.8214285714285714
```

Feature Scaling

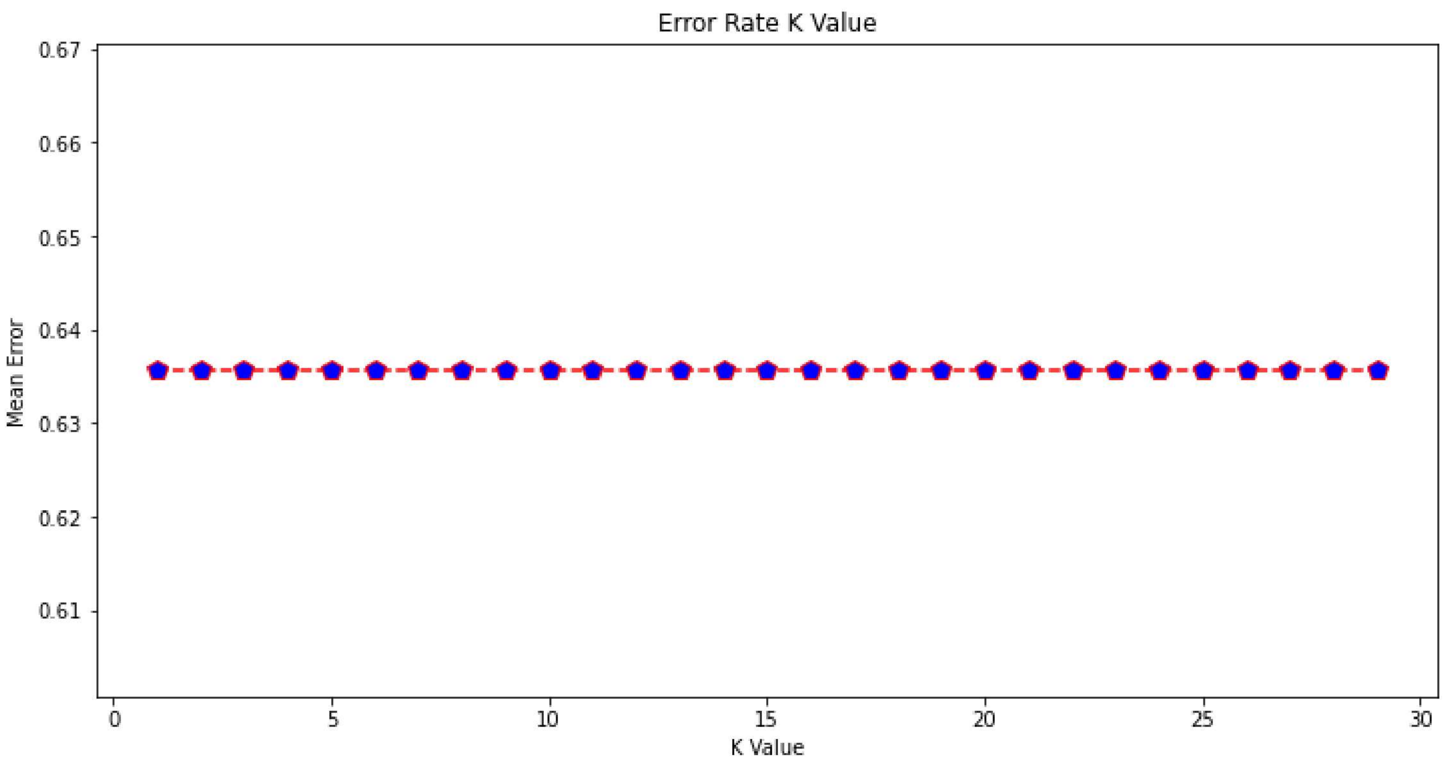
In [25]:
```
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_testb = sc.transform(X_test)
```

In [32]:
```
error = []
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

#calculating errorv for K values between 1 and 48
for i in range(1,30):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train, y_train)
    pred_i = model.predict(X_test)
    error.append(np.mean(pred_i !=y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1,30),error,color='red',linestyle='dashed',marker='p',markerfacecolor='blue',markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[32]:
```
Text(0, 0.5, 'Mean Error')
```



In [27]:
```
data.head()
```

Out[27]:

	Age	Salary	Purchase Iphone
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [28]:
```
Age = int(input("Enter New person Age:"))
Salary = int(input("Enter new person salary:"))
newperson = [[Age,Salary]]
result = model.predict(sc.transform(newperson))
print(result)

if result == 1:
    print("person might Purchase Iphone")
else:
    print("person might not purchase Iphone")
```

```
Enter New person Age:22
Enter new person salary:122
[0]
person might not purchase Iphone
```

In [29]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)

print("Accuracy of the Model: {0}%".format(accuracy_score(y_test, y_pred)*100))
```

```
Confusion Matrix:
[[78 11]
 [14 37]]
Accuracy of the Model: 82.14285714285714%
```

by using Minkowski distance methods

In [30]:
```
import pandas as pd
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: