In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv('stroke_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_lev
0	Male	58.0	1	0	Yes	Private	Urban	87.9
1	Female	70.0	0	0	Yes	Private	Rural	69.0
2	Female	52.0	0	0	Yes	Private	Urban	77.5
3	Female	75.0	0	1	Yes	Self-employed	Rural	243.5
4	Female	32.0	0	0	Yes	Private	Rural	77.6

In [4]:
```
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
df['gender'].unique()
```

Out[5]:
```
array(['Male', 'Female'], dtype=object)
```

In [6]:
```
label_encoder=LabelEncoder()
```

In [7]:
```
df['gender_label'] = label_encoder.fit_transform(df['gender'])
```

In [8]:
```
df['smoking_label'] = label_encoder.fit_transform(df['smoking_status'])
```

In [9]:
```
df['marriage_label'] = label_encoder.fit_transform(df['ever_married'])
```

In [10]:
```
df['residence_label'] = label_encoder.fit_transform(df['Residence_type'])
```

In [11]:
```
df['work_label'] = label_encoder.fit_transform(df['work_type'])
```

In [12]:
```
df.head()
```

Out[12]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_lev
0	Male	58.0	1	0	Yes	Private	Urban	87.9
1	Female	70.0	0	0	Yes	Private	Rural	69.0
2	Female	52.0	0	0	Yes	Private	Urban	77.5
3	Female	75.0	0	1	Yes	Self-employed	Rural	243.5
4	Female	32.0	0	0	Yes	Private	Rural	77.6

In [13]:
```
df.drop(['gender','smoking_status','ever_married','Residence_type','work_type'],axis=1
```

Out[13]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender_label	smoking_label
0	58.0	1	0	87.96	39.2	0	1	1
1	70.0	0	0	69.04	35.9	0	0	0
2	52.0	0	0	77.59	17.7	0	0	0
3	75.0	0	1	243.53	27.0	0	0	1
4	32.0	0	0	77.67	32.3	0	0	2
...	...	...	...	...	...	...	...	...
29060	10.0	0	0	58.64	20.4	0	0	1
29061	56.0	0	0	213.61	55.4	0	0	0
29062	82.0	1	0	91.94	28.9	0	0	0
29063	40.0	0	0	99.16	33.2	0	1	1
29064	82.0	0	0	79.48	20.6	0	0	1

29065 rows × 11 columns

Feature Selection

In [14]:
```
x = df.drop(['stroke','gender','smoking_status','ever_married','Residence_type','work_
y = df['stroke']
```

Train Test Split

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=
```

In [17]:
```
x_train[:5]
```

Out[17]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	gender_label	smoking_label	marriage
5112	79.0	1	0	66.83	19.8	0	1	
13166	32.0	0	0	110.63	33.1	1	1	
5197	64.0	0	0	109.51	25.4	0	1	
25891	24.0	0	0	95.93	23.6	1	2	
755	56.0	0	1	64.66	26.7	0	0	

In [18]:
```
y_train[:5]
```

Out[18]:
```
5112     0
13166    0
5197     0
25891    0
755      0
Name: stroke, dtype: int64
```

Logistic Regression

Model

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
log_model = LogisticRegression(max_iter=1000)
```

In [21]:
```
log_model.fit(x_train, y_train)
```

Out[21]:
```
LogisticRegression(max_iter=1000)
```

In [22]:
```
log_pred = log_model.predict(x_test)
```

Accuracy

In [23]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [24]:
```
print(confusion_matrix(y_test, log_pred))
```

```
[[8547    0]
 [ 173    0]]
```

In [25]:
```
print(classification_report(y_test, log_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8547
           1       0.00      0.00      0.00       173

    accuracy                           0.98      8720
   macro avg       0.49      0.50      0.49      8720
weighted avg       0.96      0.98      0.97      8720
```

```
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Accuracy: correct predictions / total predictions

Precision: true positive / total predicted positive (abilty of model to identify relevant data points)

Recall: true positive / total actual positive (abilty of model to final all relevant cases in dataset)

f1-score: 2 * (precision * recall) / (precision + recall) (harmonic mean of precision and recall)

Decision Tree

In [26]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [27]:
```
dtree_model = DecisionTreeClassifier()
```

In [28]:
```
dtree_model.fit(x_train, y_train)
```

Out[28]:
```
DecisionTreeClassifier()
```

In [29]:
```
dtree_pred = dtree_model.predict(x_test)
```

In [30]:
```
print(confusion_matrix(y_test, dtree_pred))
```

```
[[8337  210]
 [ 165    8]]
```

In [31]:
```
print(classification_report(y_test, dtree_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      8547
           1       0.04      0.05      0.04       173

    accuracy                           0.96      8720
   macro avg       0.51      0.51      0.51      8720
weighted avg       0.96      0.96      0.96      8720
```

Random Forest

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [33]:
```
rfc_model = RandomForestClassifier(n_estimators=1000)
```

In [34]:
```
rfc_model.fit(x_train, y_train)
```

Out[34]:
```
RandomForestClassifier(n_estimators=1000)
```

In [35]:
```
rfc_pred = rfc_model.predict(x_test)
```

In [36]:
```
print(confusion_matrix(y_test, rfc_pred))
```

```
[[8546    1]
 [ 173    0]]
```

In [37]:
```
print(classification_report(y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8547
           1       0.00      0.00      0.00       173

    accuracy                           0.98      8720
   macro avg       0.49      0.50      0.49      8720
weighted avg       0.96      0.98      0.97      8720
```

In [38]:
```
from sklearn.metrics import accuracy_score
```

In [39]:
```
accuracy_score(y_test, rfc_pred)
```

Out[39]:
```
0.9800458715596331
```

In [40]:
```
ar = [LogisticRegression(), DecisionTreeClassifier(), RandomForestClassifier(n_estimat

for i in ar:
    i.fit(x_train, y_train)
    pred = i.predict(x_test)
    print(i, "->", accuracy_score(y_test, pred))
```

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarn
ing: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

LogisticRegression() -> 0.9801605504587156
DecisionTreeClassifier() -> 0.9592889908256881
RandomForestClassifier(n_estimators=1000) -> 0.9800458715596331
```

K-Nearest Neighbors

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [42]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
```

In [43]:
```
knn_model.fit(x_train, y_train)
```

Out[43]:
```
KNeighborsClassifier()
```

In [44]:
```
knn_pred = knn_model.predict(x_test)
```

In [45]:
```
print(confusion_matrix(y_test, knn_pred))
```

```
[[8545    2]
 [ 172    1]]
```

In [46]:
```
print(classification_report(y_test, knn_pred))
```

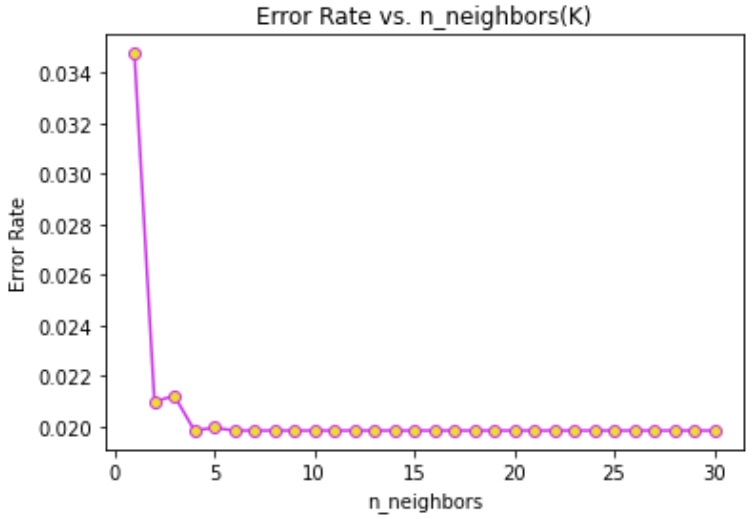
```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8547
           1       0.33      0.01      0.01       173

    accuracy                           0.98      8720
   macro avg       0.66      0.50      0.50      8720
weighted avg       0.97      0.98      0.97      8720
```

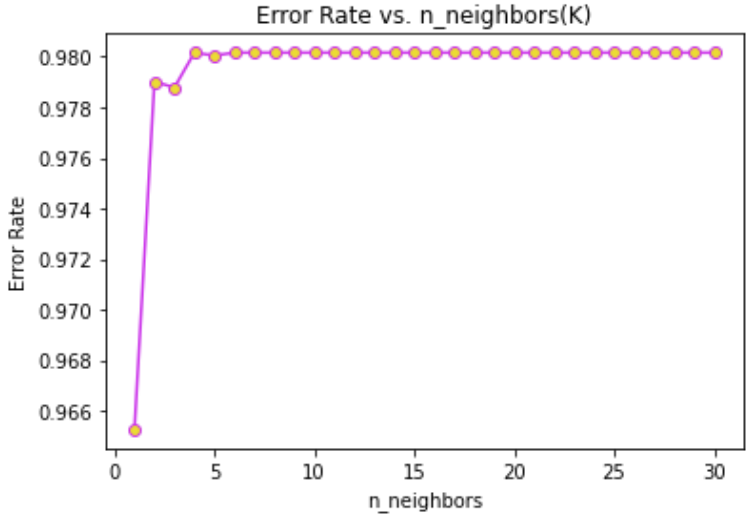
In [47]:
```
error = []
for i in range(1, 31):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    knn_model.fit(x_train, y_train)
    knn_pred = knn_model.predict(x_test)
    error.append(np.mean(knn_pred != y_test))
```

In [48]:
```
plt.plot(range(1, 31), error, color='#cc34eb', linestyle='solid', marker='o', markerfa
plt.title('Error Rate vs. n_neighbors(K)')
plt.xlabel('n_neighbors')
plt.ylabel('Error Rate')
plt.show()
```



In [49]:
```
error = []
for i in range(1, 31):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    knn_model.fit(x_train, y_train)
    knn_pred = knn_model.predict(x_test)
    error.append(np.mean(accuracy_score(y_test, knn_pred)))

plt.plot(range(1, 31), error,color='#cc34eb', linestyle='solid', marker='o', markerfac
plt.title('Error Rate vs. n_neighbors(K)')
plt.xlabel('n_neighbors')
plt.ylabel('Error Rate')
plt.show()
```



In [50]:
```
knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(x_train, y_train)
knn_pred = knn_model.predict(x_test)
```

In [51]:
```
print(confusion_matrix(y_test, knn_pred))
```

```
[[8547    0]
 [ 173    0]]
```

In [52]:
```
print(classification_report(y_test, knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8547
           1       0.00      0.00      0.00       173

    accuracy                           0.98      8720
   macro avg       0.49      0.50      0.49      8720
weighted avg       0.96      0.98      0.97      8720
```

```
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Support Vector Classifier

In [53]:
```
from sklearn.svm import SVC
```

In [54]:
```
sv_model = SVC()
```

In [55]:
```
sv_model.fit(x_train, y_train)
```

Out[55]:
```
SVC()
```

In [56]:
```
sv_pred = sv_model.predict(x_test)
```

In [57]:
```
print(confusion_matrix(y_test, sv_pred))
```

```
[[8547    0]
 [ 173    0]]
```

In [58]:
```
print(classification_report(y_test, sv_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8547
           1       0.00      0.00      0.00       173

    accuracy                           0.98      8720
   macro avg       0.49      0.50      0.49      8720
weighted avg       0.96      0.98      0.97      8720
```

```
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Final Model

In [60]:
```
knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(x, y)
knn_pred = knn_model.predict(x)
```

In [61]:
```
print(confusion_matrix(y, knn_pred))
```

```
[[28517     0]
 [  547     1]]
```

In [62]:
```
print(classification_report(y, knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     28517
           1       1.00      0.00      0.00       548

    accuracy                           0.98     29065
   macro avg       0.99      0.50      0.50     29065
weighted avg       0.98      0.98      0.97     29065
```

In [ ]:
```

```