>Beginner Level Task...

Task-1 Iris Flowers Classification ML Project :

This particular ML project is usually referred to as the “Hello World” of Machine Learning. The iris flowers dataset contains numeric attributes, and it is perfect for beginners to learn about supervised ML algorithms, mainly how to load and handle data. Also, since this is a small dataset, it can easily fit in memory without requiring special transformations or scaling capabilities.

Dataset: http://archive.ics.uci.edu/ml/datasets/Iris

Name: Manish Singh

1. Importing Required Libraries

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
print(" All required packages included successfully!")
```

```
All required packages included successfully!
```

2. Importing the Dataset

In [4]:
```
df = pd.read_csv('D:\Data_Set\Iris.csv')
df
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

3. Data Exploration

In [9]:
```
r,c = df.shape
print("Number of rows    = ",r)
print("Number of columns = ",c)
```

```
Number of rows    =  150
Number of columns =  5
```

In [5]:
```
df.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:
```
# to display stats about data
df.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:
```
# To display basic data
df.info()
```

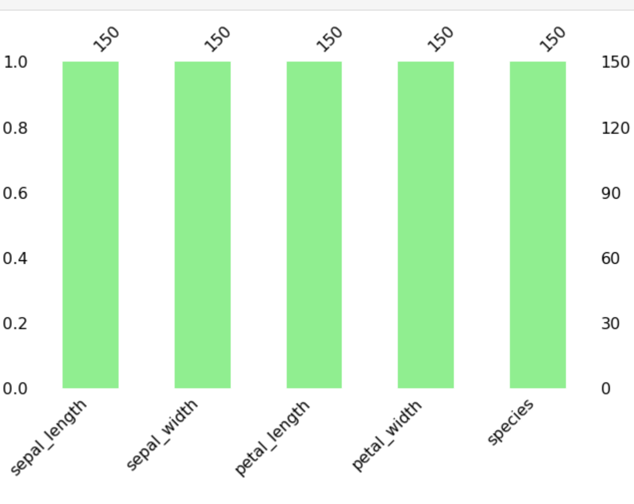
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

4. Checking Missing Values

In [10]:
```
print("Are there any missing values in the dataset ?",df.isnull().values.any())
```

```
Are there any missing values in the dataset ? False
```

In [14]:
```
msno.bar(df,figsize=(10,6),color='lightgreen')
plt.show()
```



5. Statistical Analysis

In [15]:
```
df.describe(include='all').T
```

Out[15]:

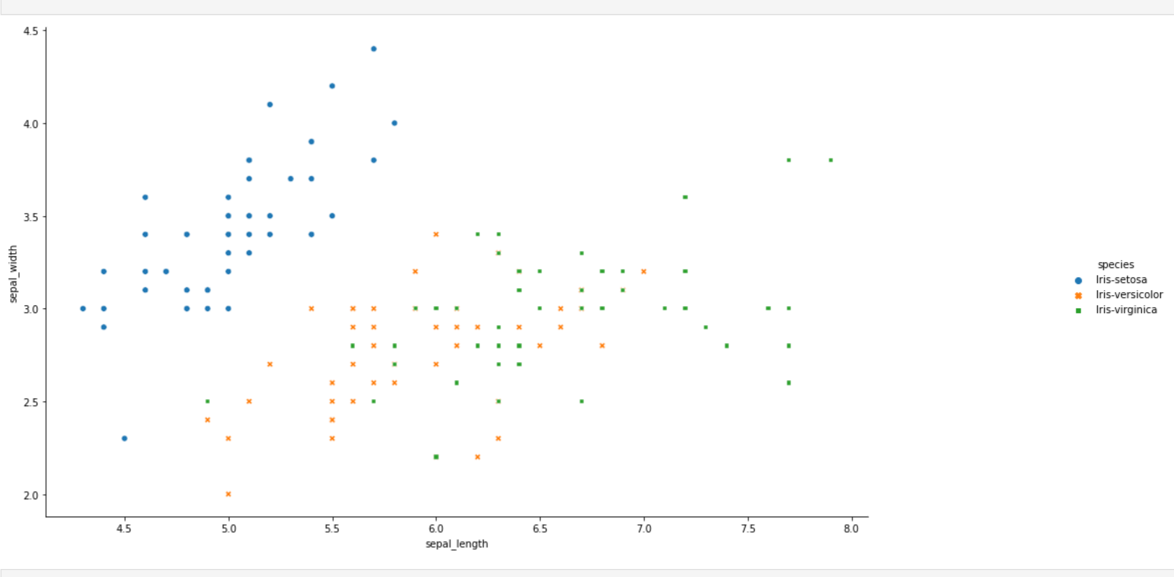
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
sepal_length	150.0	NaN	NaN	NaN	5.843333	0.828066	4.3	5.1	5.8	6.4	7.9
sepal_width	150.0	NaN	NaN	NaN	3.054	0.433594	2.0	2.8	3.0	3.3	4.4
petal_length	150.0	NaN	NaN	NaN	3.758667	1.76442	1.0	1.6	4.35	5.1	6.9
petal_width	150.0	NaN	NaN	NaN	1.198667	0.763161	0.1	0.3	1.3	1.8	2.5
species	150	3	Iris-setosa	50	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [16]:
```
df['species'].unique()
```

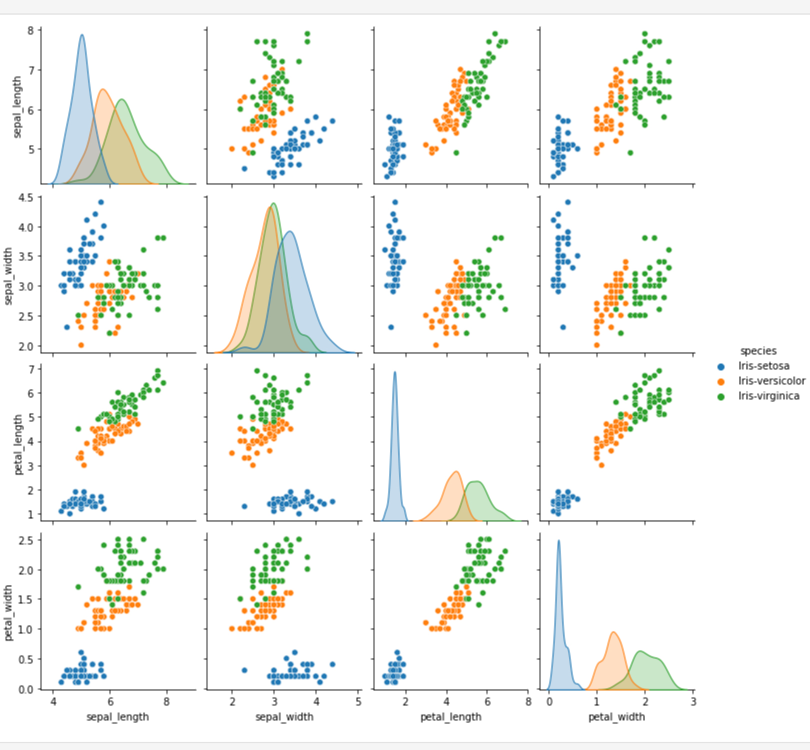
Out[16]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

6. Parametric Visualization

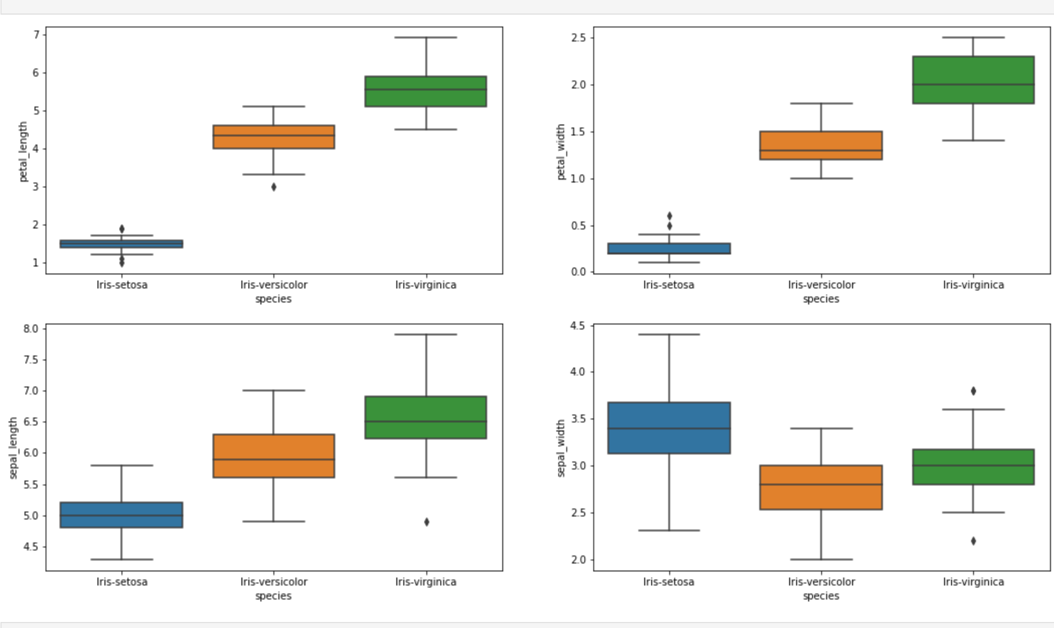
In [17]:
```
g=sns.relplot(x='sepal_length',y='sepal_width',data=df,hue='species',style='species')
g.fig.set_size_inches(18,8)
plt.show()
```



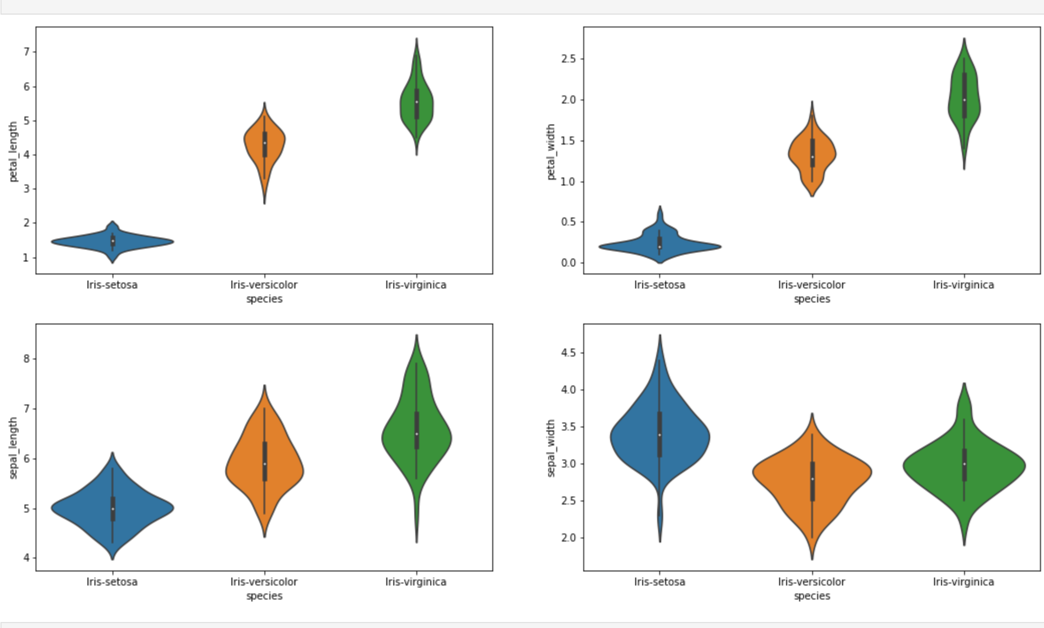
In [18]:
```
sns.pairplot(df,hue='species')
plt.show()
```



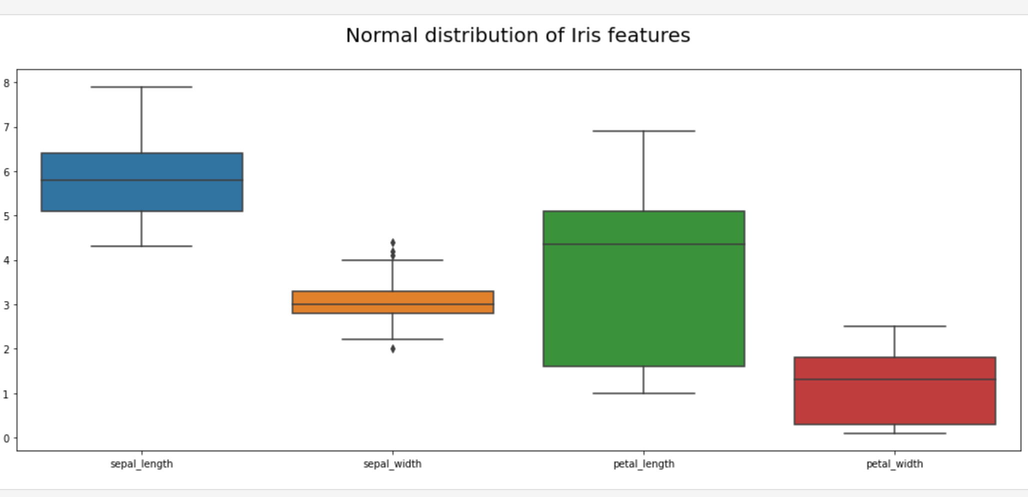
In [19]:
```
plt.figure(figsize=(18,10))
plt.subplot(2,2,1)
sns.boxplot(x='species',y='petal_length',data=df)
plt.subplot(2,2,2)
sns.boxplot(x='species',y='petal_width',data=df)
plt.subplot(2,2,3)
sns.boxplot(x='species',y='sepal_length',data=df)
plt.subplot(2,2,4)
sns.boxplot(x='species',y='sepal_width',data=df)
plt.show()
```



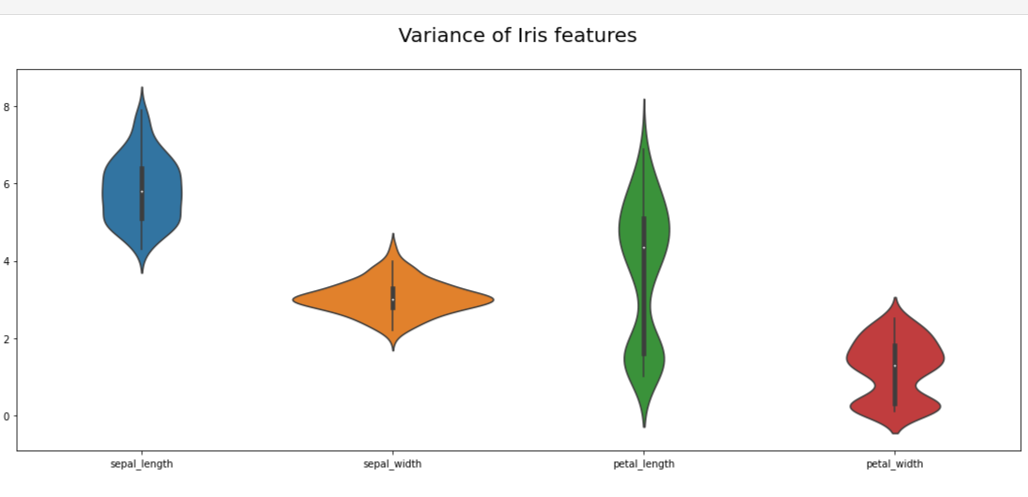
In [20]:
```
plt.figure(figsize=(18,10))
plt.subplot(2,2,1)
sns.violinplot(x='species',y='petal_length',data=df)
plt.subplot(2,2,2)
sns.violinplot(x='species',y='petal_width',data=df)
plt.subplot(2,2,3)
sns.violinplot(x='species',y='sepal_length',data=df)
plt.subplot(2,2,4)
sns.violinplot(x='species',y='sepal_width',data=df)
plt.show()
```



In [21]:
```
plt.figure(figsize=(18,7))
sns.boxplot(data=df).set_title("Normal distribution of Iris features\n",size=20)
plt.show()
```

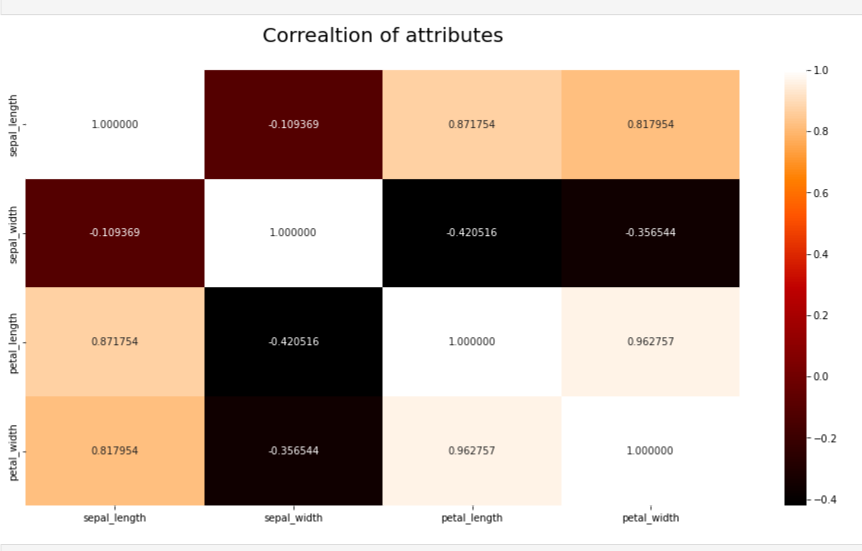


In [22]:
```
plt.figure(figsize=(18,7))
sns.violinplot(data=df).set_title("Variance of Iris features\n",size=20)
plt.show()
```



7. Attribute Correlation

In [23]:
```
plt.figure(figsize=(16,8))
sns.heatmap(df.corr(),annot=True,fmt='f',cmap='gist_heat').set_title('Correaltion of attributes\n',size=20)
plt.show()
```



In [24]:
```
X = df.iloc[:,0:4].values
y = df.iloc[:,4].values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

8. Metric

In [27]:
```
from sklearn.metrics import make_scorer,accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
print("All necessary metrics included!")
```

```
All necessary metrics included!
```

9.Model Selection

In [28]:
```
from sklearn.model_selection import KFold,train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
print("All Machine Learning packages included!")
```

```
All Machine Learning packages included!
```

10.Random Forest

In [29]:
```
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
acc_rf = round(accuracy_score(y_test,y_pred)*100,2)
rf_acc = round(rf.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of Random Forest\n",cm)
print("Accuracy of Random Forest = ",acc)
print("Precision of Random Forest = ",prec)
print("Recall of Random Forest = ",recall)
print("f1 score of Random Forest = ",f1)
```

```
Confusion matrix of Random Forest
 [[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy of Random Forest =  1.0
Precision of Random Forest =  1.0
Recall of Random Forest =  1.0
f1 score of Random Forest =  1.0
```

11.Logistic Regression

In [30]:
```
lg = LogisticRegression(solver='lbfgs',max_iter=400)
lg.fit(X_train,y_train)
y_pred = lg.predict(X_test)
acc_lg = round(accuracy_score(y_test,y_pred)*100,2)
lg_acc = round(lg.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of Logistic Regression\n",cm)
print("Accuracy of Logistic Regression = ",acc)
print("Precision of Logistic Regression = ",prec)
print("Recall of Logistic Regression = ",recall)
print("f1 score of Logistic Regression = ",f1)
```

```
Confusion matrix of Logistic Regression
 [[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy of Logistic Regression =  1.0
Precision of Logistic Regression =  1.0
Recall of Logistic Regression =  1.0
f1 score of Logistic Regression =  1.0
```

12.K Nearest Neighbours

In [31]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
acc_knn = round(accuracy_score(y_test,y_pred)*100,2)
knn_acc = round(knn.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of K Nearest Neighbour\n",cm)
print("Accuracy of K Nearest Neighbour = ",acc)
print("Precision of K Nearest Neighbour = ",prec)
print("Recall of K Nearest Neighbour = ",recall)
print("f1 score of K Nearest Neighbour = ",f1)
```

```
Confusion matrix of K Nearest Neighbour
 [[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
Accuracy of K Nearest Neighbour =  0.9666666666666667
Precision of K Nearest Neighbour =  0.9666666666666667
Recall of K Nearest Neighbour =  0.9666666666666667
f1 score of K Nearest Neighbour =  0.9666666666666667
```

13.KNN

In [32]:
```
plt.figure(figsize=(20,7))
a_index = list(range(1,50))
a = pd.Series()
x = range(1,50)
for i in list(range(1,50)):
  model = KNeighborsClassifier(n_neighbors=i)
  model.fit(X_train,y_train)
  prediction = model.predict(X_test)
  a = a.append(pd.Series(accuracy_score(y_test,prediction)))
plt.plot(a_index,a,marker="*")
plt.xticks(x)
plt.show()
```

```
C:\Users\Lenovo\AppData\Local\Temp/ipykernel_6860/4153686925.py:3: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning.
  a = pd.Series()
```

14.Gaussian Naive Bayes

In [33]:
```
gauss = GaussianNB()
gauss.fit(X_train,y_train)
y_pred = gauss.predict(X_test)
acc_gauss = round(accuracy_score(y_test,y_pred)*100,2)
gauss_acc = round(gauss.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of K Nearest Neighbour\n",cm)
print("Accuracy of K Nearest Neighbour = ",acc)
print("Precision of K Nearest Neighbour = ",prec)
print("Recall of K Nearest Neighbour = ",recall)
print("f1 score of K Nearest Neighbour = ",f1)
```

```
Confusion matrix of K Nearest Neighbour
 [[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
Accuracy of K Nearest Neighbour =  0.9666666666666667
Precision of K Nearest Neighbour =  0.9666666666666667
Recall of K Nearest Neighbour =  0.9666666666666667
f1 score of K Nearest Neighbour =  0.9666666666666667
```

14.Linear Support Vector Classifier

In [34]:
```
lsvc = LinearSVC(max_iter=4000)
lsvc.fit(X_train,y_train)
y_pred = lsvc.predict(X_test)
acc_lsvc = round(accuracy_score(y_test,y_pred)*100,2)
lsvc_acc = round(lsvc.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of K Nearest Neighbour\n",cm)
print("Accuracy of K Nearest Neighbour = ",acc)
print("Precision of K Nearest Neighbour = ",prec)
print("Recall of K Nearest Neighbour = ",recall)
print("f1 score of K Nearest Neighbour = ",f1)
```

```
Confusion matrix of K Nearest Neighbour
 [[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy of K Nearest Neighbour =  1.0
Precision of K Nearest Neighbour =  1.0
Recall of K Nearest Neighbour =  1.0
f1 score of K Nearest Neighbour =  1.0
```

15.Decision Tree Classifier

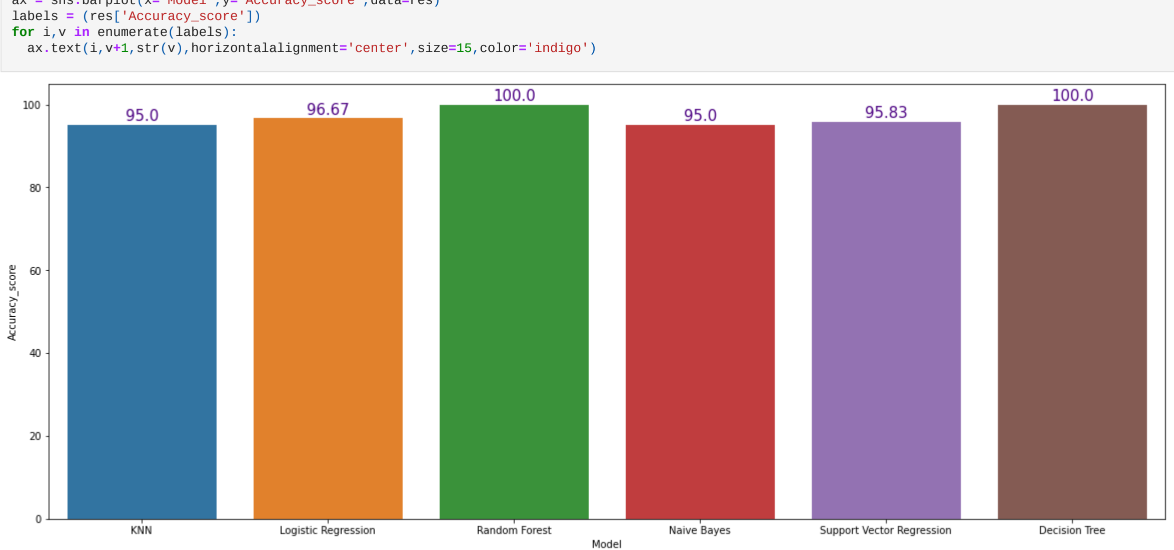
In [35]:
```
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
acc_dt = round(accuracy_score(y_test,y_pred)*100,2)
dt_acc = round(dt.score(X_train,y_train)*100,2)
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred,average='micro')
recall = recall_score(y_test,y_pred,average='micro')
f1 = f1_score(y_test,y_pred,average='micro')
print("Confusion matrix of K Nearest Neighbour\n",cm)
print("Accuracy of K Nearest Neighbour = ",acc)
print("Precision of K Nearest Neighbour = ",prec)
print("Recall of K Nearest Neighbour = ",recall)
print("f1 score of K Nearest Neighbour = ",f1)
```

```
Confusion matrix of K Nearest Neighbour
 [[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy of K Nearest Neighbour =  1.0
Precision of K Nearest Neighbour =  1.0
Recall of K Nearest Neighbour =  1.0
f1 score of K Nearest Neighbour =  1.0
```

17.Model Scorer

In [36]:
```
res = pd.DataFrame(
    {
        'Model':['KNN','Logistic Regression','Random Forest','Naive Bayes','Support Vector Regression','Decision Tree'],
     'Score':[acc_knn,acc_lg,acc_rf,acc_gauss,acc_lsvc,acc_dt],
     'Accuracy_score':[knn_acc,lg_acc,rf_acc,gauss_acc,lsvc_acc,dt_acc]
    }
)


res
plt.figure(figsize=(20,8))
ax = sns.barplot(x='Model',y='Accuracy_score',data=res)
labels = (res['Accuracy_score'])
for i,v in enumerate(labels):
  ax.text(i,v+1,str(v),horizontalalignment='center',size=15,color='indigo')
```



THANK YOU !