In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression , LinearRegression
from sklearn.metrics import accuracy_score , classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import warnings
```

In [2]:
```
data= pd.read_csv("C:\\Users\\AMIT\\Downloads\\data (2).csv")
```

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

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 33 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [5]:
```
data.isna().sum()
```

Out[5]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

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

Out[6]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400

8 rows × 32 columns

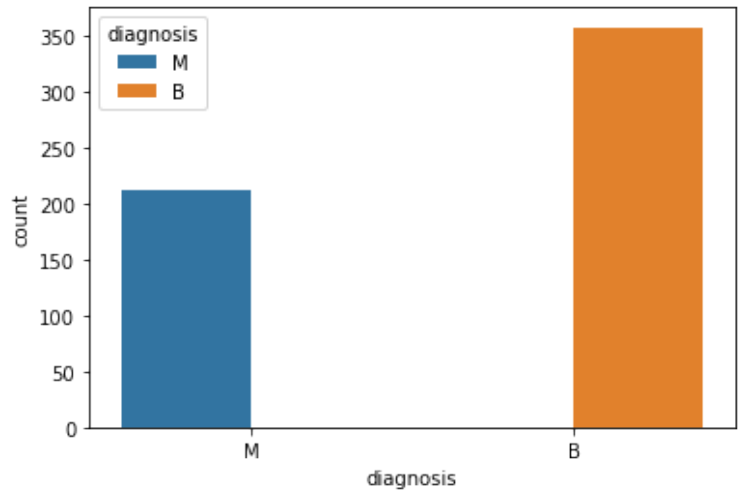
In [7]:
```
df=data.drop(columns='Unnamed: 32')
```

In [8]:
```
df.shape
```

Out[8]:
```
(569, 32)
```

In [9]:
```
sns.countplot(x='diagnosis' , hue='diagnosis', data=df)
```

Out[9]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```

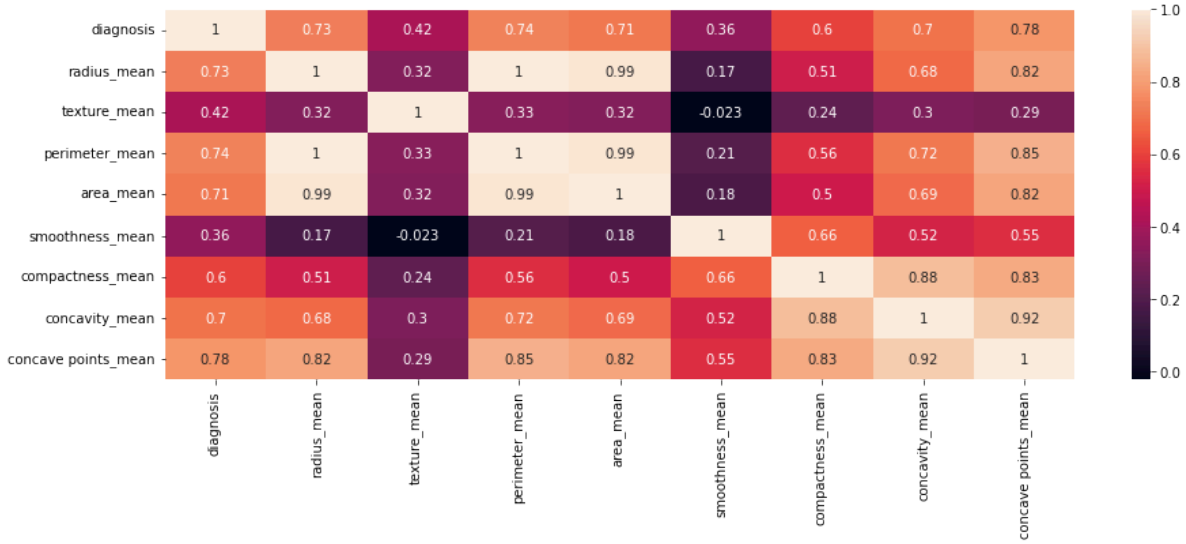


In [10]:
```
lb= LabelEncoder()
```

In [11]:
```
df.iloc[:, 1]=lb.fit_transform(df.iloc[:, 1].values)
```

In [12]:
```
plt.figure(figsize=(15,5))
sns.heatmap(df.iloc[:, 1:10].corr(), annot=True)
```

Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
X= df.iloc[:, 2:32].values
y=df.iloc[:, 1].values
```

In [14]:
```
X_train, X_test,y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=0
```

In [15]:
```
st=StandardScaler()
X_train=st.fit_transform(X_train)
X_test=st.fit_transform(X_test)
```

In [16]:
```
data.drop(['Unnamed: 32'], axis=1, inplace=True)
y = data['diagnosis'].map({'M': 1, 'B': 0})
X = data.drop(['id','diagnosis'], axis =1)
```

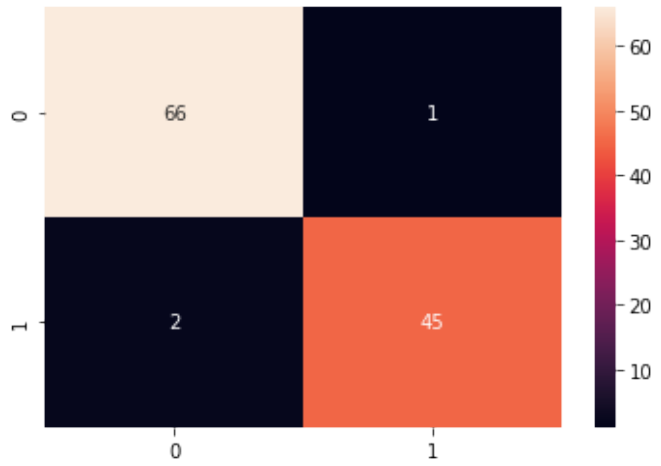
In [17]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initialize Random Forest classifier
rf = RandomForestClassifier()

# Train the Random Forest classifier
rf.fit(X_train, y_train)

# Check the accuracy on the training set
train_accuracy = rf.score(X_train, y_train)

# Make predictions on the test set and calculate accuracy
test_predictions = rf.predict(X_test)
test_accuracy = accuracy_score(y_test, test_predictions)

# Print classification report
print(classification_report(y_test, test_predictions))

print("Training Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        67
           1       0.98      0.96      0.97        47

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Training Accuracy: 1.0
Test Accuracy: 0.9736842105263158
```

In [18]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, test_predictions)
cm
```

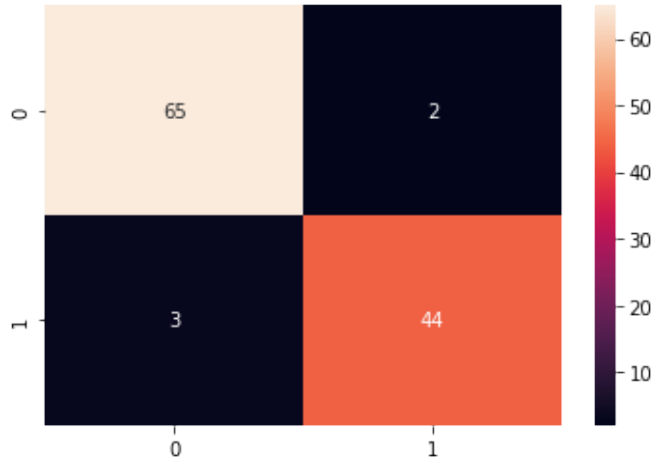
Out[18]:
```
array([[66,  1],
       [ 2, 45]], dtype=int64)
```

In [19]:
```
sns.heatmap(cm, annot=True)
plt.show()
```



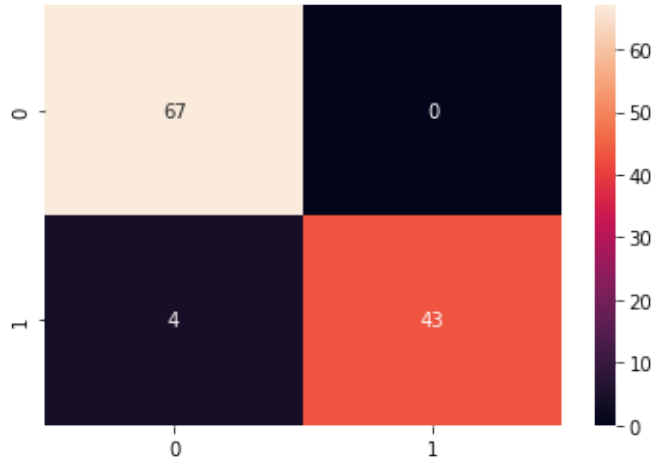
In [20]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print("Logistic Regression Accuracy:", accuracy_logreg)
cm_logreg = confusion_matrix(y_test, y_pred_logreg)
sns.heatmap(cm_logreg, annot=True)
plt.show()
```

```
Logistic Regression Accuracy: 0.956140350877193
```



In [21]:
```
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("K-Nearest Neighbors Accuracy:", accuracy_knn)
cm_knn = confusion_matrix(y_test, y_pred_knn)
sns.heatmap(cm_knn, annot=True)
plt.show()
```

```
K-Nearest Neighbors Accuracy: 0.9649122807017544
```

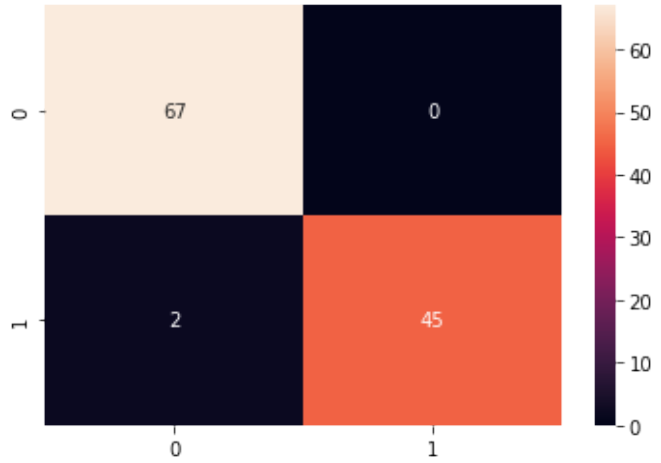


In [22]:
```
# Linear Regression
linear_regression = LinearRegression()
linear_regression.fit(X_train, y_train)
y_pred_linear_regression = linear_regression.predict(X_test)
mse_linear_regression = mean_squared_error(y_test, y_pred_linear_regression)
print("Linear Regression MSE:", mse_linear_regression)
```

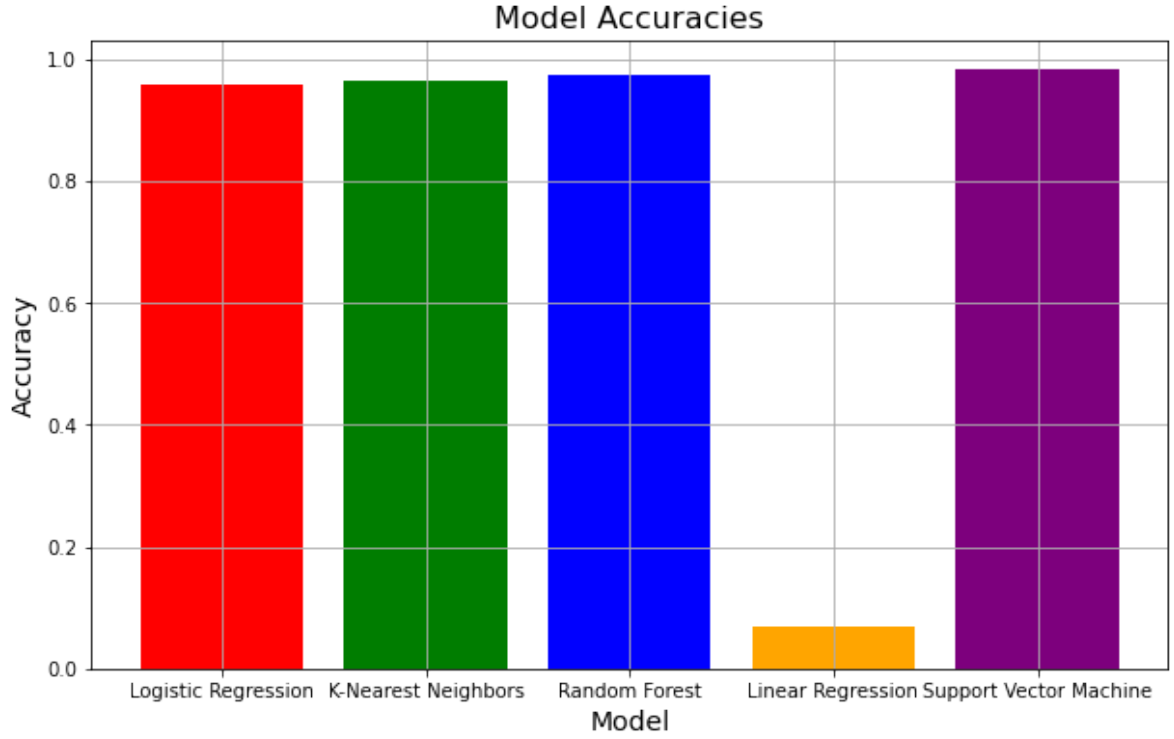
```
Linear Regression MSE: 0.068366677209251
```

In [23]:
```
# Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("Support Vector Machine Accuracy:", accuracy_svm)
cm_svm = confusion_matrix(y_test, y_pred_svm)
sns.heatmap(cm_svm, annot=True)
plt.show()
```

```
Support Vector Machine Accuracy: 0.9824561403508771
```



In [24]:
```
import matplotlib.pyplot as plt
# Create a bar chart with different colors for each bar
colors = ["red", "green", "blue", "orange", "purple"]
labels = ["Logistic Regression", "K-Nearest Neighbors", "Random Forest", "Linear
accuracies = [accuracy_logreg, accuracy_knn, test_accuracy, mse_linear_regression

plt.figure(figsize=(10, 6))
plt.bar(labels, accuracies, color=colors)

# Add title and axis labels
plt.title("Model Accuracies", fontsize=16)
plt.xlabel("Model", fontsize=14)
plt.ylabel("Accuracy", fontsize=14)

# Add grid and show plot
plt.grid(True)
plt.show()
```



In [26]:
```
# Take user input for new data
new_data = []
print("Enter the values for each feature:")

# Prompt the user for each feature value
for col in X.columns:
    val = input(f"Enter {col}: ")
    new_data.append(float(val))

# Convert user input into numpy array
new_data = np.array(new_data).reshape(1, -1)

# Standardize the new data using the same scaler
new_data = st.transform(new_data)

# Predict using the trained Random Forest classifier
prediction = rf.predict(new_data)

# Display the prediction
if prediction[0] == 1:
    print("The diagnosis is Malignant (M)")
else:
    print("The diagnosis is Benign (B)")
```

```
Enter the values for each feature:
Enter radius_mean: 567
Enter texture_mean: 67.8
Enter perimeter_mean: 76
Enter area_mean: 45
Enter smoothness_mean: 5656
Enter compactness_mean: 57
Enter concavity_mean: 788
Enter concave points_mean: 67
Enter symmetry_mean: 566
Enter fractal_dimension_mean: 46
Enter radius_se: 45
Enter texture_se: 56
Enter perimeter_se: 90
Enter area_se: 89.6
Enter smoothness_se: 0.86
Enter compactness_se: 676
Enter concavity_se: 67
Enter concave points_se: 432
Enter symmetry_se: 35.9
Enter fractal_dimension_se: 87.0
Enter radius_worst: 7.8
Enter texture_worst: 7.09
Enter perimeter_worst: 0.98
Enter area_worst: 87
Enter smoothness_worst: 6
Enter compactness_worst: 6
Enter concavity_worst: 7
Enter concave points_worst: 65
Enter symmetry_worst: 557
Enter fractal_dimension_worst: 6768
The diagnosis is Benign (B)
```