Import required libraries

In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:
```
df = pd.read_csv('insurance2.csv')
```

In [15]:
```
df.head()
```

Out[15]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             1338 non-null   int64  
 1   sex             1338 non-null   int64  
 2   bmi             1338 non-null   float64
 3   children        1338 non-null   int64  
 4   smoker          1338 non-null   int64  
 5   region          1338 non-null   int64  
 6   charges         1338 non-null   float64
 7   insuranceclaim  1338 non-null   int64  
dtypes: float64(2), int64(6)
memory usage: 83.8 KB
```

In [17]:
```
df['region'].value_counts().sort_values()
```

Out[17]:
```
0    324
3    325
1    325
2    364
Name: region, dtype: int64
```

In [18]:
```
df['children'].value_counts().sort_values()
```

Out[18]:
```
5     18
4     25
3    157
2    240
1    324
0    574
Name: children, dtype: int64
```

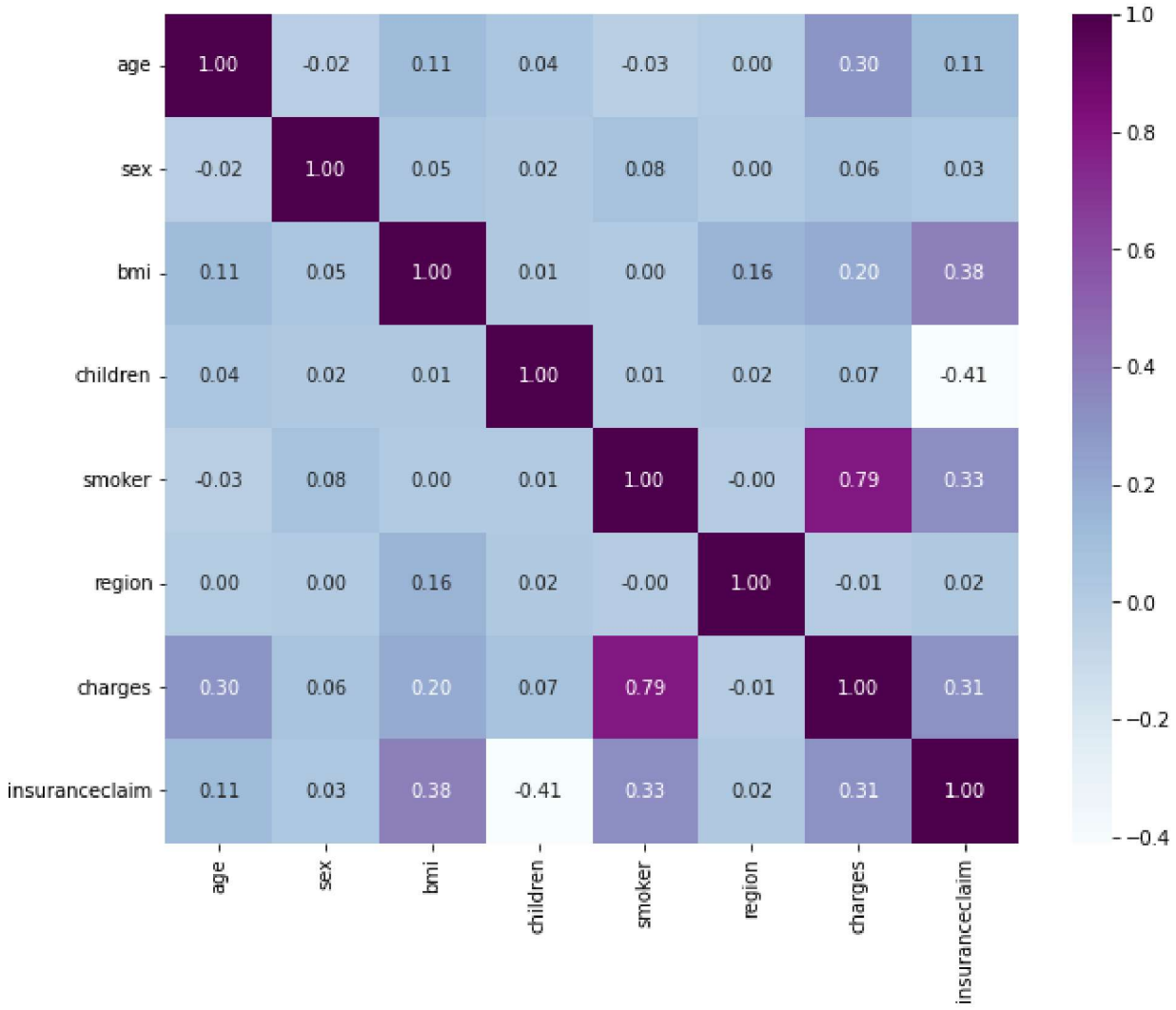
In [19]:
```
df_copy = df.copy()
df_copy.replace(inplace=True)
```

In [20]:
```
df_copy.describe()
```

Out[20]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265	0.585202
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237	0.492871
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900	0.000000
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150	0.000000
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000	1.000000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515	1.000000
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010	1.000000

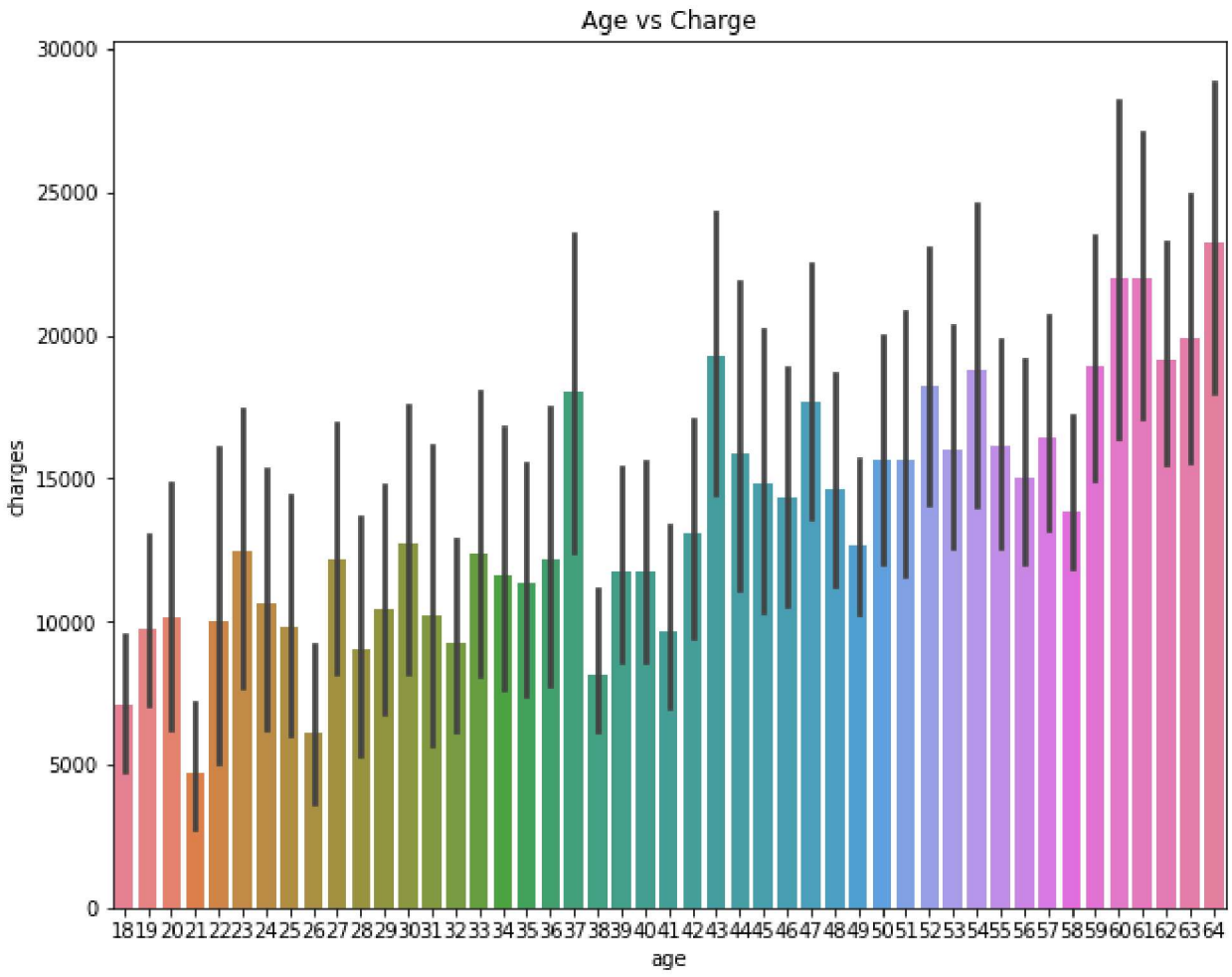
In [21]:
```
corr = df_copy.corr()
fig, ax = plt.subplots(figsize=(10,8))
sns.heatmap(corr,cmap='BuPu', annot=True, fmt=".2f", ax=ax)
plt.show()
```



Find the correlation of every pair of features

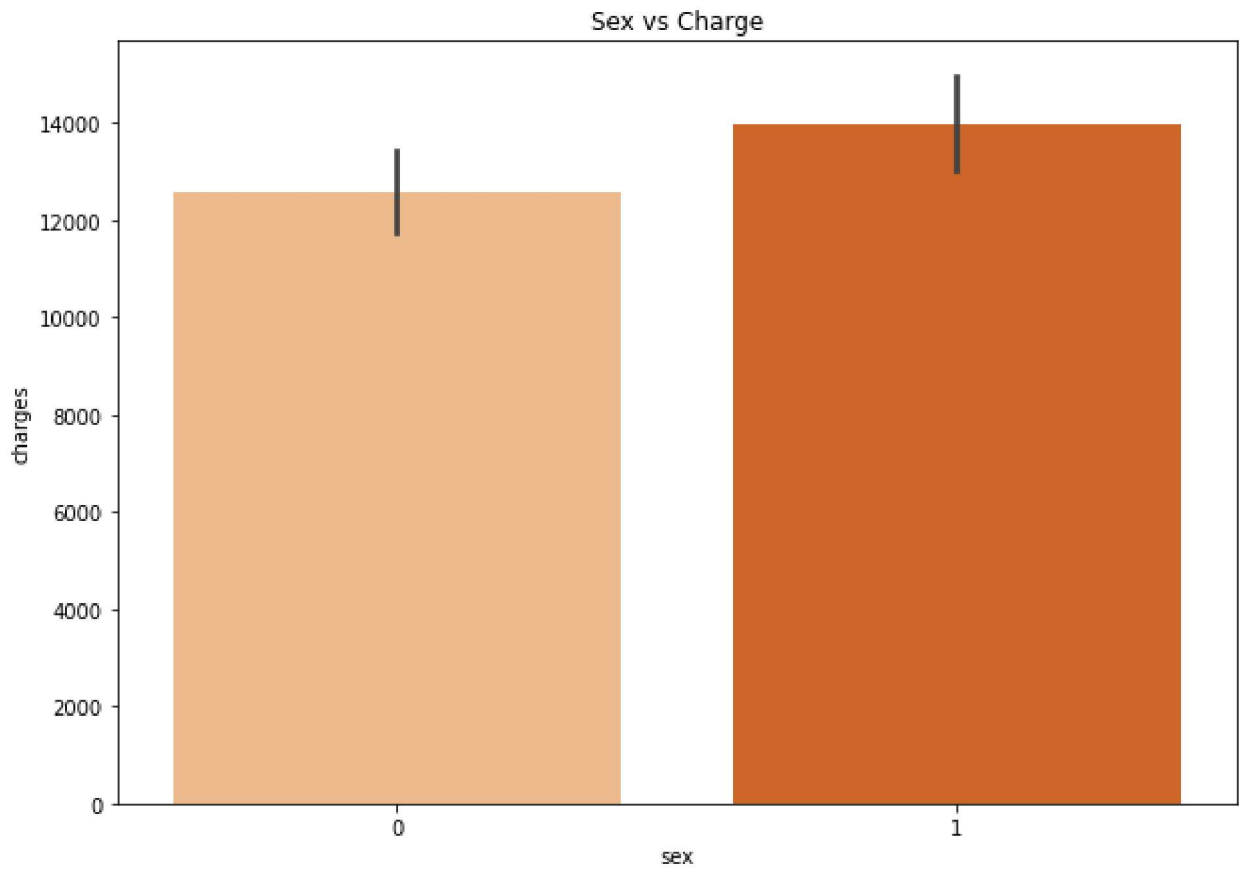
In [22]:
```
plt.figure(figsize=(10,8))
plt.title('Age vs Charge')
sns.barplot(x='age',y='charges',data=df_copy,palette='husl')
```

Out[22]: <AxesSubplot:title={'center':'Age vs Charge'}, xlabel='age', ylabel='charges'>



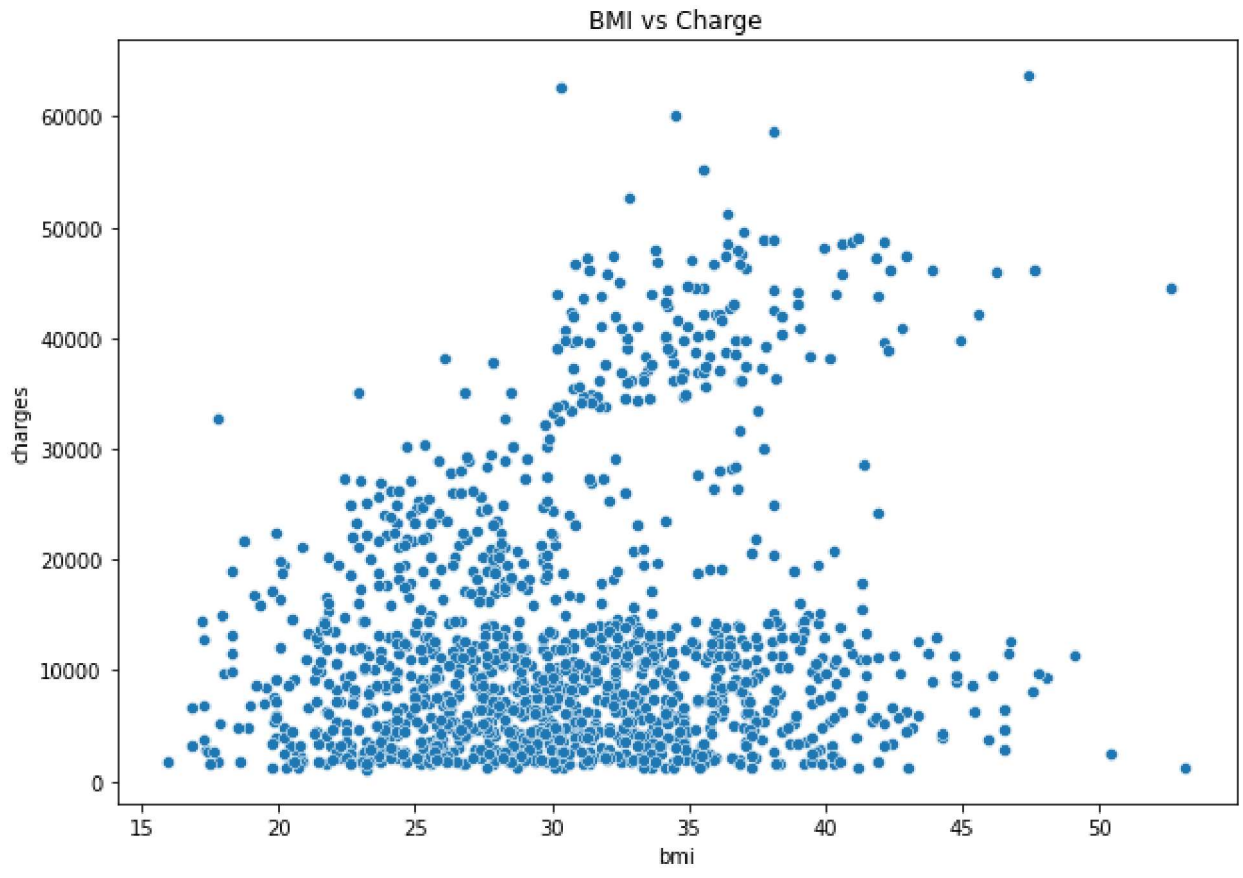
In [23]:
```
plt.figure(figsize=(10,7))
plt.title('Sex vs Charge')
sns.barplot(x='sex',y='charges',data=df_copy,palette='Oranges')
```

Out[23]: <AxesSubplot:title={'center':'Sex vs Charge'}, xlabel='sex', ylabel='charges'>



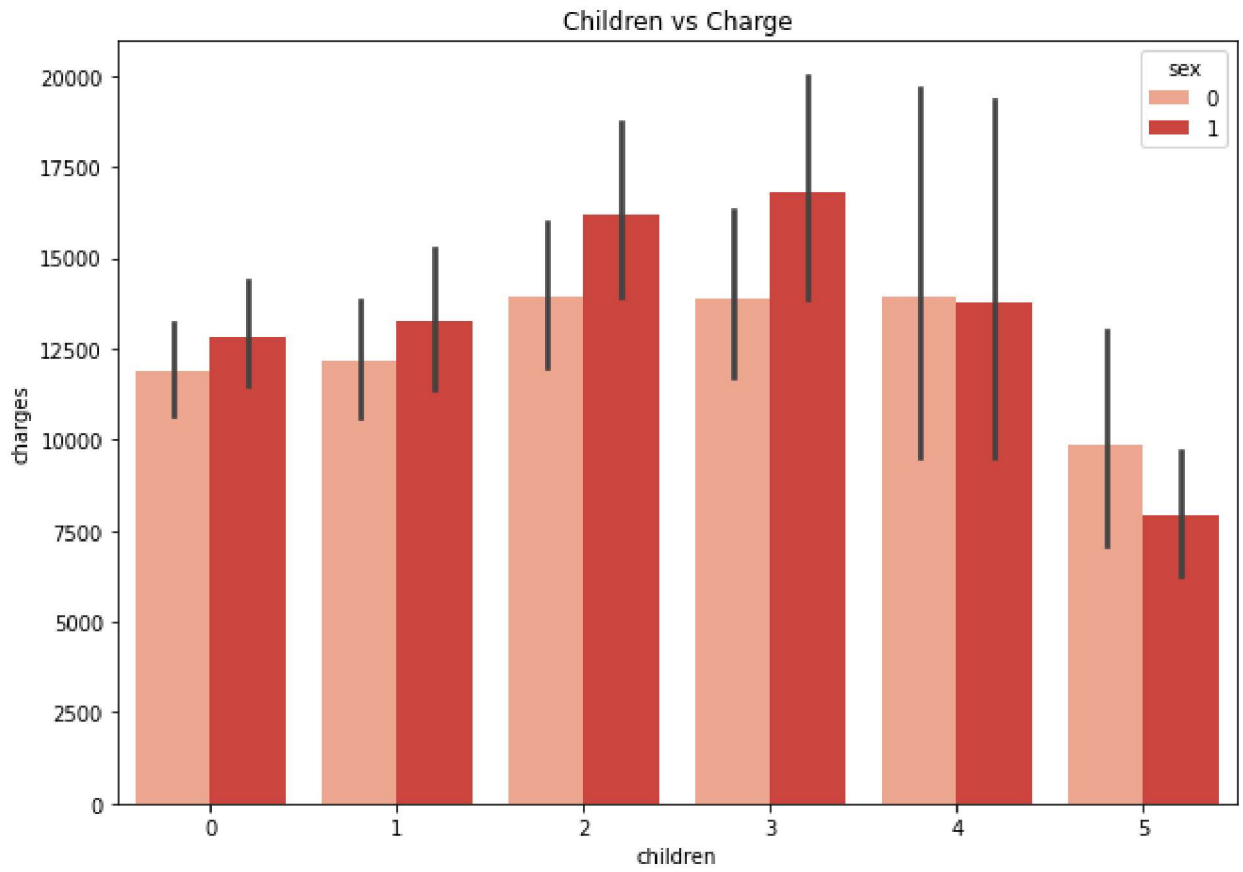
In [24]:
```
plt.figure(figsize=(10,7))
plt.title('BMI vs Charge')
sns.scatterplot(x='bmi',y='charges',data=df_copy,palette='Blues')
```

Out[24]: <AxesSubplot:title={'center':'BMI vs Charge'}, xlabel='bmi', ylabel='charges'>



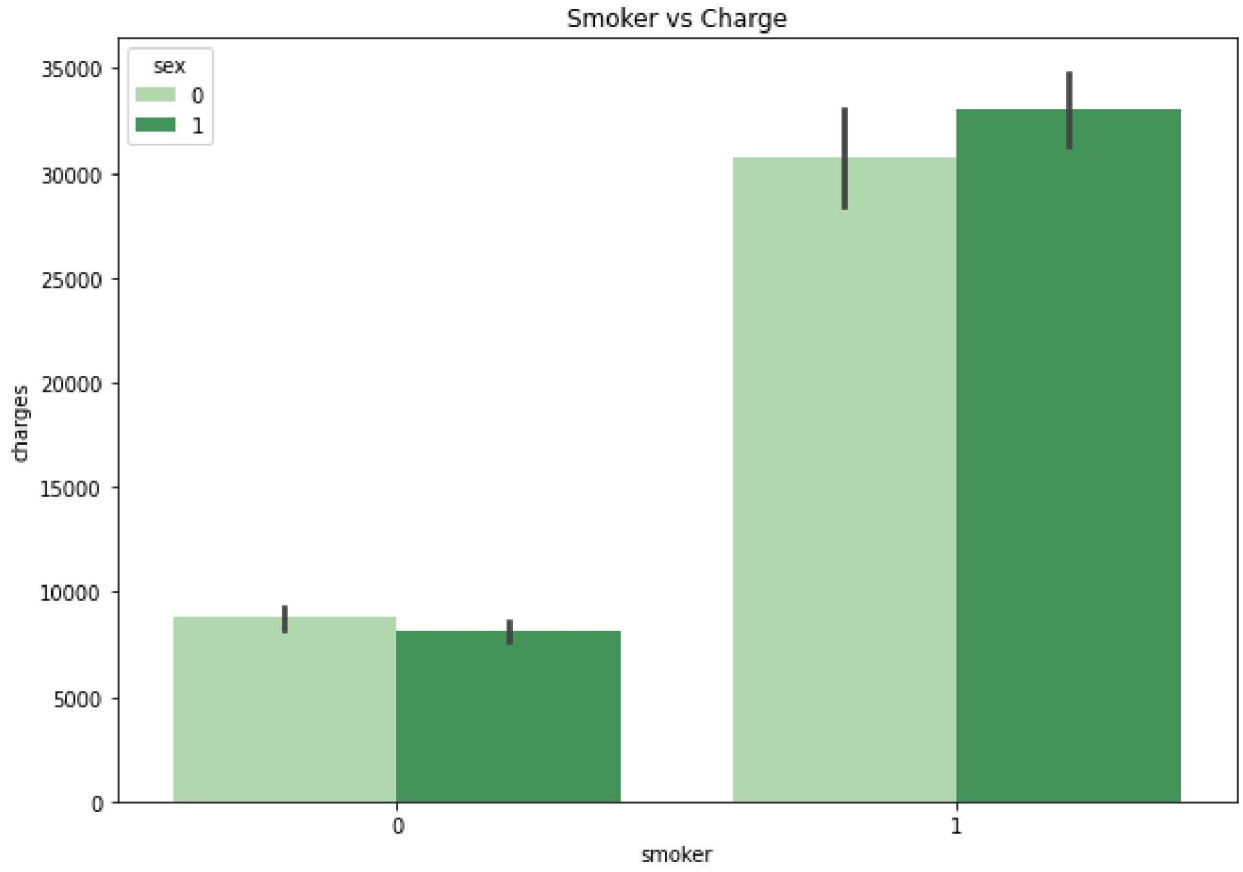
In [25]:
```
plt.figure(figsize=(10,7))
plt.title('Children vs Charge')
sns.barplot(x='children',y='charges',hue='sex',data=df_copy,palette='Reds')
```

Out[25]: <AxesSubplot:title={'center':'Children vs Charge'}, xlabel='children', ylabel='charges'>



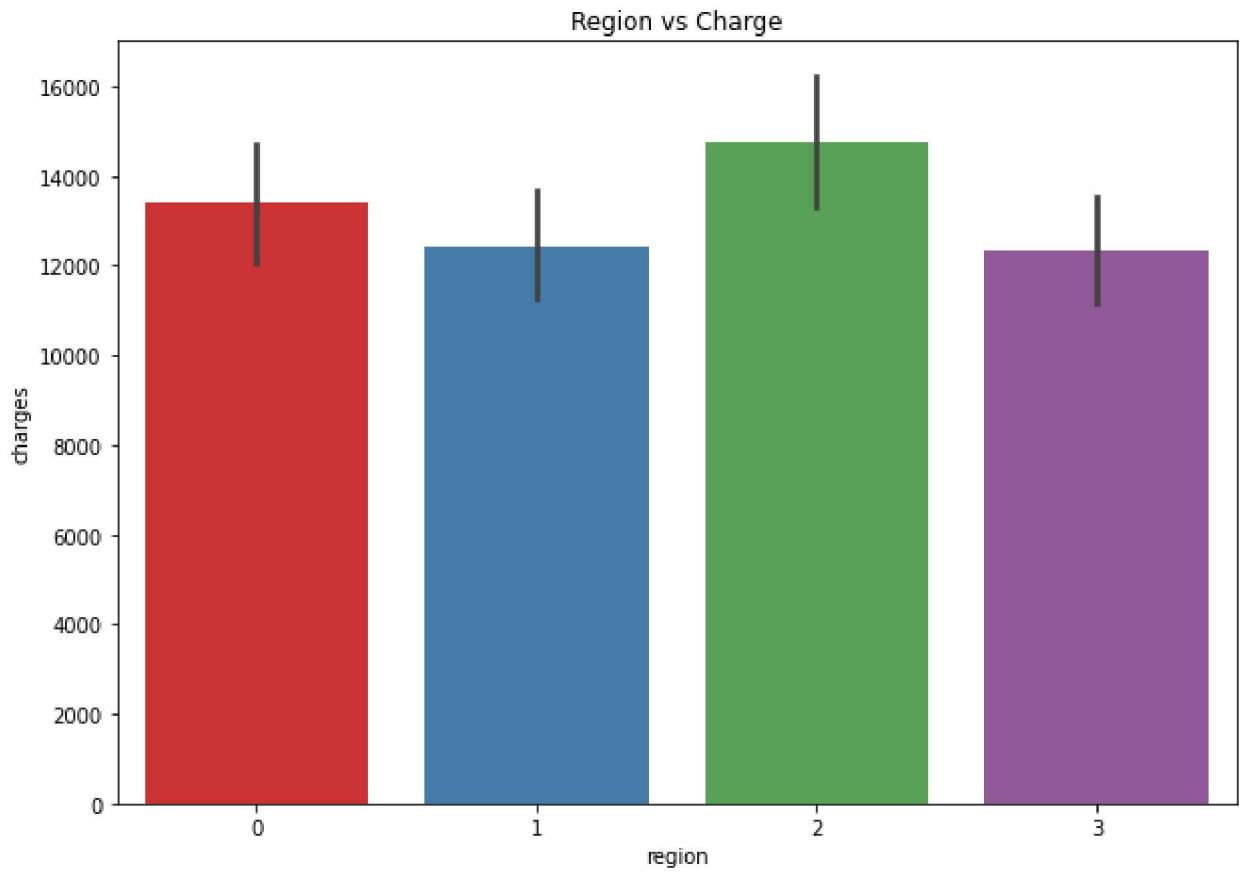
In [26]:
```
plt.figure(figsize=(10,7))
plt.title('Smoker vs Charge')
sns.barplot(x='smoker',y='charges',hue='sex',data=df_copy,palette='Greens')
```

Out[26]: <AxesSubplot:title={'center':'Smoker vs Charge'}, xlabel='smoker', ylabel='charges'>



In [27]:
```
plt.figure(figsize=(10,7))
plt.title('Region vs Charge')
sns.barplot(x='region',y='charges',data=df_copy, palette='Set1')
```

Out[27]: <AxesSubplot:title={'center':'Region vs Charge'}, xlabel='region', ylabel='charges'>

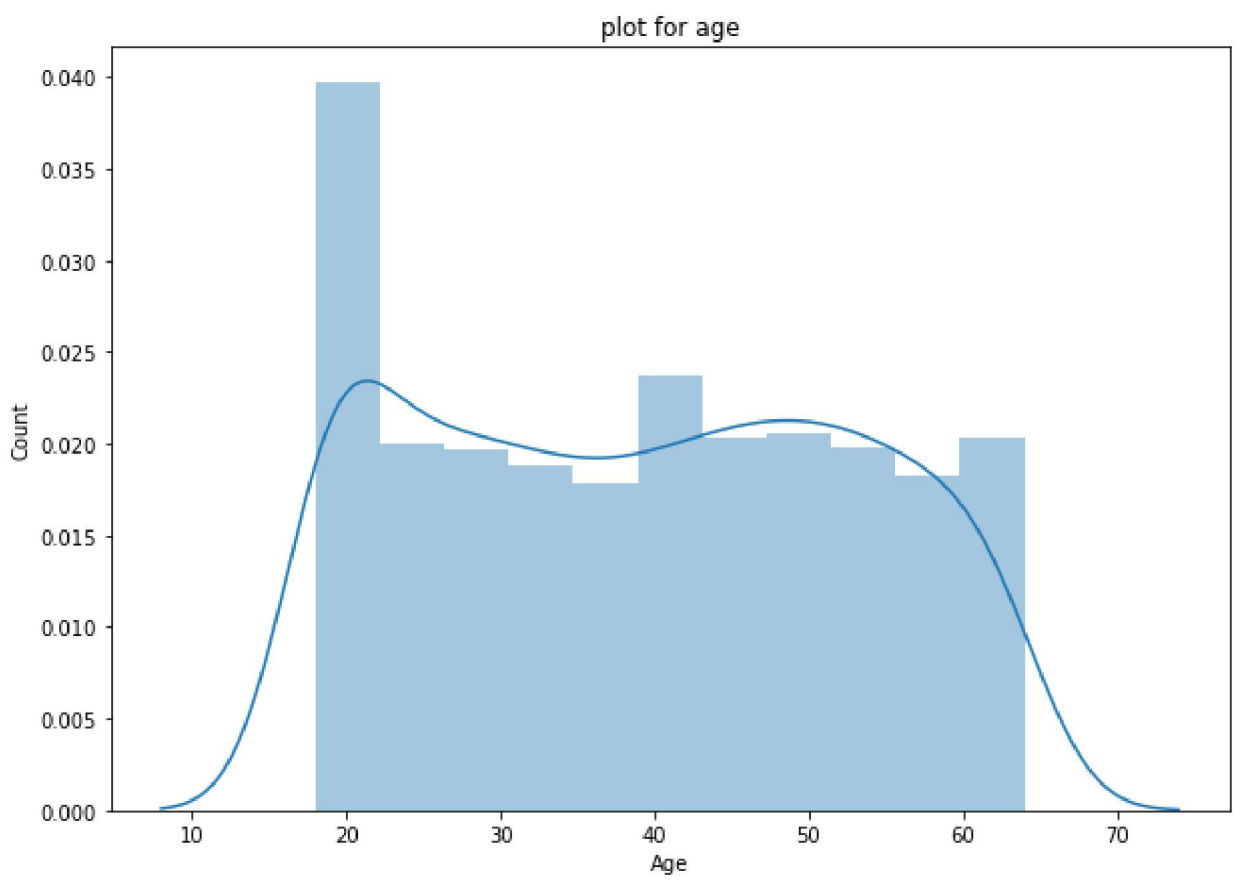


Visualize the correlations using a heatmap

In [28]:
```
plt.figure(figsize=(10,7))
sns.distplot(df_copy['age'])
plt.title('plot for age')
plt.xlabel('Age')
plt.ylabel('Count')
```

```
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

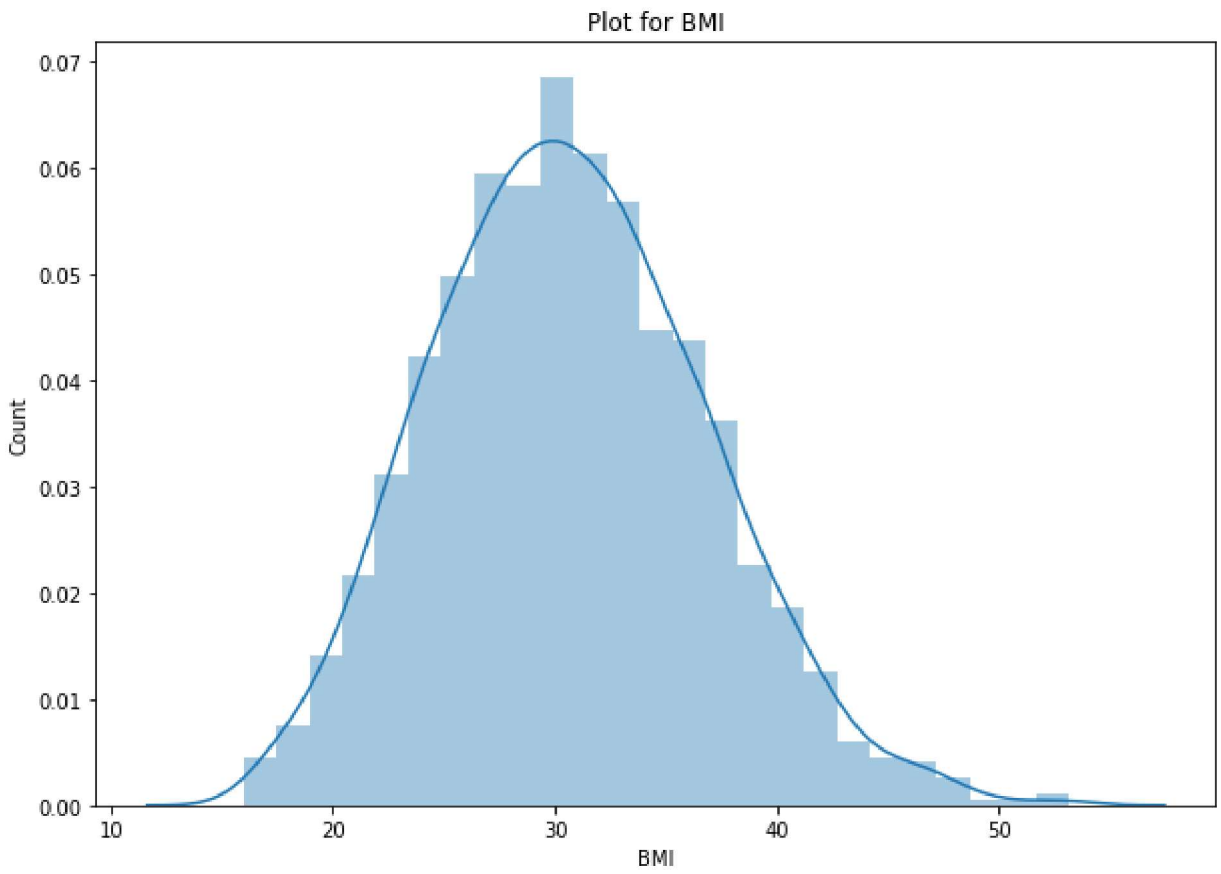
Out[28]: Text(0, 0.5, 'Count')



In [29]:
```
plt.figure(figsize=(10,7))
sns.distplot(df_copy['bmi'])
plt.title('Plot for BMI')
plt.xlabel('BMI')
plt.ylabel('Count')
```

```
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

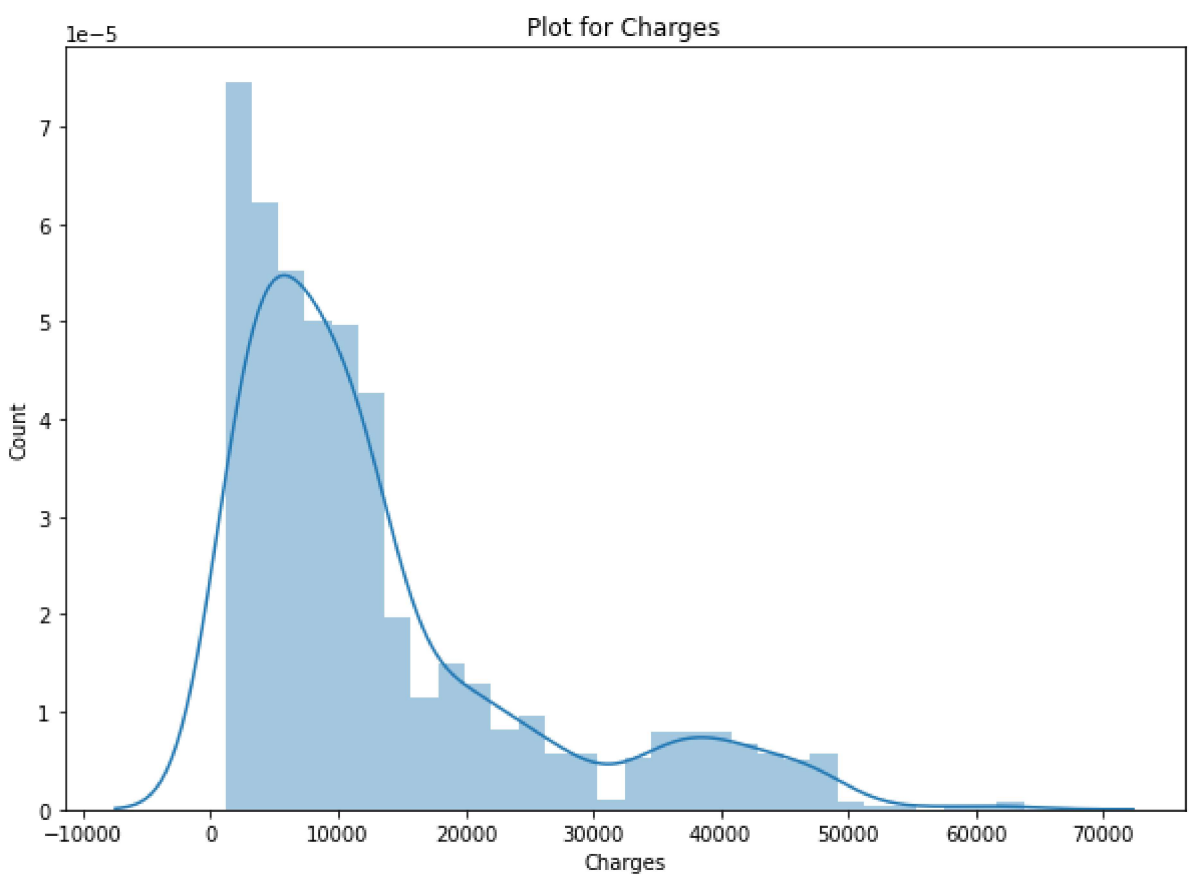
Out[29]: Text(0, 0.5, 'Count')



In [30]:
```
plt.figure(figsize=(10,7))
sns.distplot(df_copy['charges'])
plt.title('Plot for Charges')
plt.xlabel('Charges')
plt.ylabel('Count')
```

```
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[30]: Text(0, 0.5, 'Count')



Normalize your inputs

In [31]:
```
from sklearn.preprocessing import StandardScaler
data_pre = df_copy.copy()

tempBmi = data_pre.bmi
tempBmi = tempBmi.values.reshape(-1,1)
data_pre['bmi'] = StandardScaler().fit_transform(tempBmi)

tempAge = data_pre.age
tempAge = tempAge.values.reshape(-1,1)
data_pre['age'] = StandardScaler().fit_transform(tempAge)

tempCharges = data_pre.charges
tempCharges = tempCharges.values.reshape(-1,1)
data_pre['charges'] = StandardScaler().fit_transform(tempCharges)
```

In [32]:
```
data_pre.head()
```

Out[32]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	-1.438764	0	-0.453320	0	1	3	0.298584	1
1	-1.509965	1	0.509621	1	0	2	-0.953689	1
2	-0.797954	1	0.383307	3	0	2	-0.728675	0
3	-0.441948	1	-1.305531	0	0	1	0.719843	0
4	-0.513149	1	-0.292556	0	0	1	-0.776802	1

In [38]:
```
X = data_pre.drop('charges',axis=1).values
y = data_pre['charges'].values.reshape(-1,1)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

print('Size of X_train : ', X_train.shape)
print('Size of y_train : ', y_train.shape)
print('Size of X_test : ', X_test.shape)
print('Size of Y_test : ', y_test.shape)
```

```
Size of X_train :  (1070, 7)
Size of y_train :  (1070, 1)
Size of X_test :  (268, 7)
Size of Y_test :  (268, 1)
```

Use the test data to find out the accuracy of the model

In [39]:
```
#Import required libraries
```

In [40]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR

from sklearn.metrics import r2_score, mean_squared_error, accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score, RandomizedSearchCV, GridSearchCV
```

In [41]:
```
pip install LinearRegression
```

```
Requirement already satisfied: LinearRegression in c:\users\harsh dhapodkar\anaconda3\lib\site-packages (0.0.
1)
Note: you may need to restart the kernel to use updated packages.
```

Linear_Regression

In [42]:
```
%%time
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
```

```
CPU times: total: 15.6 ms
Wall time: 19 ms
```

Out[42]:
```
LinearRegression()
```

In [44]:
```
cv_linear_reg = cross_val_score(estimator = linear_reg, X=X, y=y, cv=10)

y_pred_linear_reg_train = linear_reg.predict(X_train)
r2_score_linear_reg_train = r2_score(y_train, y_pred_linear_reg_train)

y_pred_linear_reg_test = linear_reg.predict(X_test)
r2_score_linear_reg_test = r2_score(y_test, y_pred_linear_reg_test)

rmse_linear = (np.sqrt(mean_squared_error(y_test, y_pred_linear_reg_test)))

print('CV Linear Regression : {0:3f}'.format(cv_linear_reg.mean()))
print('R2_score (train) : {0:3f}'.format(r2_score_linear_reg_train))
print('R2_score (test) : {0:3f}',format(r2_score_linear_reg_test))
print('RMSE : {0:3f}'.format(rmse_linear))
```

```
CV Linear Regression : 0.746173
R2_score (train) : 0.744378
R2_score (test) : {0:3f} 0.7824434217148324
RMSE : 0.480085
```

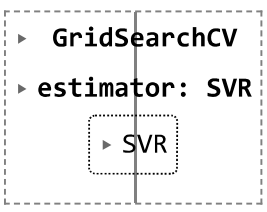
Support_Vector_Regression

In [56]:
```
X_c = df_copy.drop('charges',axis=1).values
y_c = df_copy['charges'].values.reshape(-1,1)

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_c,y_c,test_size=0.2, random_state=42)

X_train_scaled = StandardScaler().fit_transform(X_train_c)
y_train_scaled = StandardScaler().fit_transform(y_train_c)
X_test_scaled = StandardScaler().fit_transform(X_test_c)
y_test_scaled = StandardScaler().fit_transform(y_test_c)

svr = SVR()
#svr.fit(X_train_scaled, y_train_scaled.ravel())
```

In [57]:
```
parameters =  { 'kernel' : ['rbf', 'sigmoid'],
                'gamma' : [0.001, 0.01, 0.1, 1, 'scale'],
                'tol' : [0.0001],
                'C': [0.001, 0.01, 0.1, 1, 10, 100] }
svr_grid = GridSearchCV(estimator=svr, param_grid=parameters, cv=10, verbose=4, n_jobs=-1)
svr_grid.fit(X_train_scaled, y_train_scaled.ravel())
```

```
Fitting 10 folds for each of 60 candidates, totalling 600 fits
```

Out[57]:



In [60]:
```
svr = SVR(C=10, gamma=0.1, tol=0.0001)
svr.fit(X_train_scaled, y_train_scaled.ravel())
print(svr_grid.best_estimator_)
print(svr_grid.best_score_)
```

```
SVR(C=1, gamma=0.1, tol=0.0001)
0.8346976763110956
```

In [61]:
```
cv_svr = svr_grid.best_score_

y_pred_svr_train = svr.predict(X_train_scaled)
r2_score_svr_train = r2_score(y_train_scaled, y_pred_svr_train)

y_pred_svr_test = svr.predict(X_test_scaled)
r2_score_svr_test = r2_score(y_test_scaled, y_pred_svr_test)

rmse_svr = (np.sqrt(mean_squared_error(y_test_scaled, y_pred_svr_test)))

print('CV : {0:.3f}'.format(cv_svr.mean()))
print('R2_score (train) : {0:.3f}'.format(r2_score_svr_train))
print('R2 score (test) : {0:.3f}'.format(r2_score_svr_test))
print('RMSE : {0:.3f}'.format(rmse_svr))
```

```
CV : 0.835
R2_score (train) : 0.858
R2 score (test) : 0.872
RMSE : 0.358
```

Ridge_Regressor

In [62]:
```
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Ridge

steps = [ ('scalar', StandardScaler()),
        ('poly', PolynomialFeatures(degree=2)),
        ('model', Ridge())]

ridge_pipe = Pipeline(steps)
```

In [63]:
```
parameters = { 'model__alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2,1,2,5,10,20,25,35, 43,55,100], 'model__random_sta
reg_ridge = GridSearchCV(ridge_pipe, parameters, cv=10)
reg_ridge = reg_ridge.fit(X_train, y_train.ravel())
```

```
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:212: LinAlgWarning: Ill-co
nditioned matrix (rcond=1.61727e-19): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

In [64]:
```
reg_ridge.best_estimator_, reg_ridge.best_score_
```

Out[64]:
```
(Pipeline(steps=[('scalar', StandardScaler()), ('poly', PolynomialFeatures()),
                 ('model', Ridge(alpha=20, random_state=42))]),
 0.8262985743663187)
```

In [65]:
```
ridge = Ridge(alpha=20, random_state=42)
ridge.fit(X_train_scaled, y_train_scaled.ravel())
cv_ridge = reg_ridge.best_score_

y_pred_ridge_train = ridge.predict(X_train_scaled)
r2_score_ridge_train = r2_score(y_train_scaled, y_pred_ridge_train)

y_pred_ridge_test = ridge.predict(X_test_scaled)
r2_score_ridge_test = r2_score(y_test_scaled, y_pred_ridge_test)

rmse_ridge = (np.sqrt(mean_squared_error(y_test_scaled, y_pred_linear_reg_test)))
print('CV : {0:.3f}'.format(cv_ridge.mean()))
print('R2 score (train) : {0:.3f}'.format(r2_score_ridge_train))
print('R2 score (test) : {0:.3f}'.format(r2_score_ridge_test))
print('RMSE : {0:.3f}'.format(rmse_ridge))
```

```
CV : 0.826
R2 score (train) : 0.744
R2 score (test) : 0.784
RMSE : 0.466
```

Random_Forest_Regressor

In [66]:
```
%%time
reg_rf = RandomForestRegressor()
parameters = { 'n_estimators':[600,1000,1200],
              'max_features': ["auto"],
              'max_depth':[40,50,60],
              'min_samples_split': [5,7,9],
              'min_samples_leaf': [7,10,12],
              'criterion': ['mse']}

reg_rf_gscv = GridSearchCV(estimator=reg_rf, param_grid=parameters, cv=10, n_jobs=-1)
reg_rf_gscv = reg_rf_gscv.fit(X_train_scaled, y_train_scaled.ravel())
```

```
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:400: FutureWarning: Criterion
'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is equi
valent.
  warn(
C:\Users\Harsh Dhapodkar\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_feat
ures='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set
`max_features=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and Ext
raTreesRegressors.
  warn(

CPU times: total: 2.38 s
Wall time: 2min 37s
```

In [67]:
```
reg_rf_gscv.best_score_, reg_rf_gscv.best_estimator_
```

Out[67]:
```
(0.8493784486574111,
 RandomForestRegressor(criterion='mse', max_depth=60, max_features='auto',
                       min_samples_leaf=12, min_samples_split=7,
                       n_estimators=600))
```

In [68]:
```
rf_reg = RandomForestRegressor(max_depth=50, min_samples_leaf=12, min_samples_split=7,
                               n_estimators=1200)
rf_reg.fit(X_train_scaled, y_train_scaled.ravel())
```

Out[68]:
```
RandomForestRegressor(max_depth=50, min_samples_leaf=12, min_samples_split=7,
                      n_estimators=1200)
```

In [69]:
```
cv_rf = reg_rf_gscv.best_score_

y_pred_rf_train = rf_reg.predict(X_train_scaled)
r2_score_rf_train = r2_score(y_train, y_pred_rf_train)

y_pred_rf_test = rf_reg.predict(X_test_scaled)
r2_score_rf_test = r2_score(y_test_scaled, y_pred_rf_test)

rmse_rf = np.sqrt(mean_squared_error(y_test_scaled, y_pred_rf_test))

print('CV : {0:.3f}'.format(cv_rf.mean()))
print('R2 score (train) : {0:.3f}'.format(r2_score_rf_train))
print('R2 score (test) : {0:.3f}'.format(r2_score_rf_test))
print('RMSE : {0:.3f}'.format(rmse_rf))
```

```
CV : 0.849
R2 score (train) : 0.886
R2 score (test) : 0.879
RMSE : 0.348
```

Visualize how your model uses the different features and which features have a greater effect

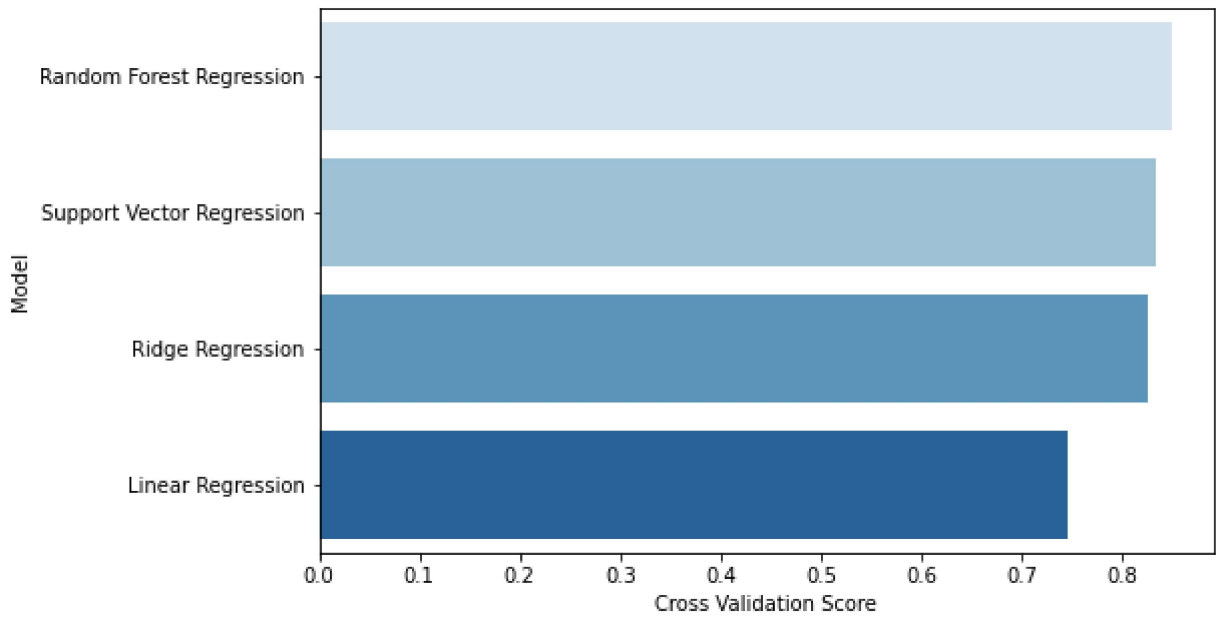
In [72]:
```
models = [('Linear Regression', rmse_linear, r2_score_linear_reg_train, r2_score_linear_reg_test, cv_linear_reg.
           ('Ridge Regression', rmse_ridge, r2_score_ridge_train, r2_score_ridge_test, cv_ridge.mean()),
           ('Support Vector Regression', rmse_svr, r2_score_svr_train, r2_score_svr_test, cv_svr.mean()),
           ('Random Forest Regression', rmse_rf, r2_score_rf_train, r2_score_rf_test, cv_rf.mean())
          ]
```

In [73]:
```
predict = pd.DataFrame(data = models, columns=['Model', 'RMSE', 'R2_Score(training)', 'R2_Score(test)', 'Cross-V
predict
```

Out[73]:

	Model	RMSE	R2_Score(training)	R2_Score(test)	Cross-Validation
0	Linear Regression	0.480085	0.744378	0.782443	0.746173
1	Ridge Regression	0.465941	0.743985	0.783995	0.826299
2	Support Vector Regression	0.357856	0.858198	0.871939	0.834698
3	Random Forest Regression	0.348095	0.885616	0.878830	0.849378

In [77]:
```
plt.figure(figsize=(8,5))
predict.sort_values(by=['Cross-Validation'], ascending=False, inplace=True)

sns.barplot(x='Cross-Validation', y='Model',data = predict, palette='Blues')
plt.xlabel('Cross Validation Score')
plt.ylabel('Model')
plt.show()
```



In [78]:
```
df_copy.head()
```

Out[78]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1