In [566]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [567]:
```
insurancedata = pd.read_csv(r'C:\Users\alharbi\Downloads\insurance.csv')
```

In [568]:
```
insurancedata.head(5)
```

Out[568]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [569]:
```
insurancedata.shape
```

Out[569]: (1338, 7)

In [570]:
```
insurancedata.describe()
```

Out[570]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [571]:
```
insurancedata.dtypes
```

Out[571]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

In [572]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x='age', y='charges', data=insurancedata)
plt.title('Regression Plot of charges by age')
```

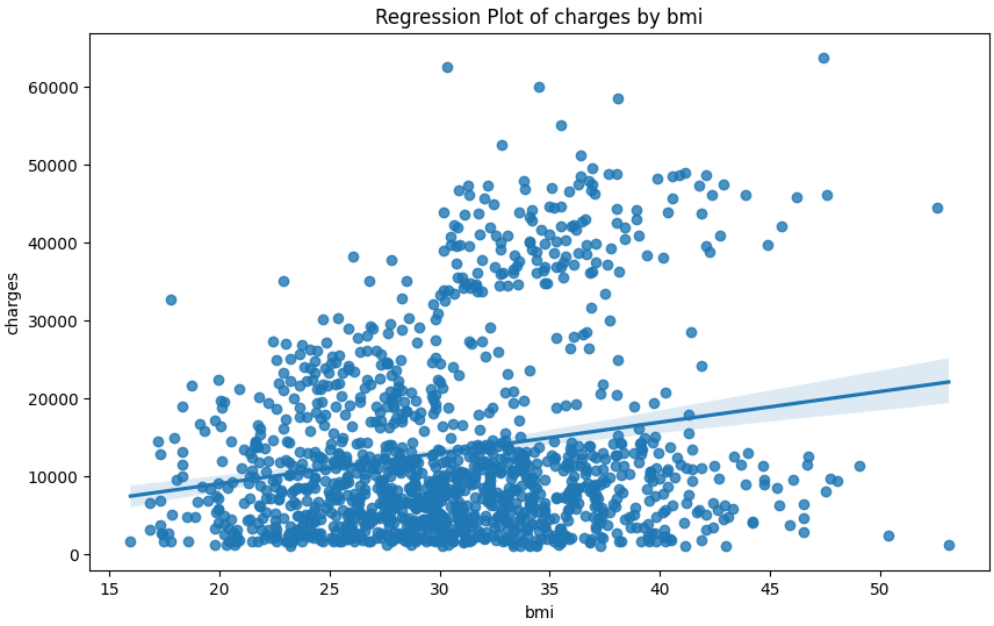
Out[572]: Text(0.5, 1.0, 'Regression Plot of charges by age')

In [573]:
```
from scipy import stats
pearson_coef_age, p_value_age = stats.pearsonr(insurancedata['age'], insurancedata['charges'])
print("The Pearson Correlation Coefficient between age and charges is", pearson_coef_age, "with a P-value of P=", p_value_age)
```

```
The Pearson Correlation Coefficient between age and charges is 0.29900819333064754 with a P-value of P= 4.886693331718298e-29
```

In [574]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x='bmi', y='charges', data=insurancedata)
plt.title('Regression Plot of charges by bmi')
```

Out[574]: Text(0.5, 1.0, 'Regression Plot of charges by bmi')

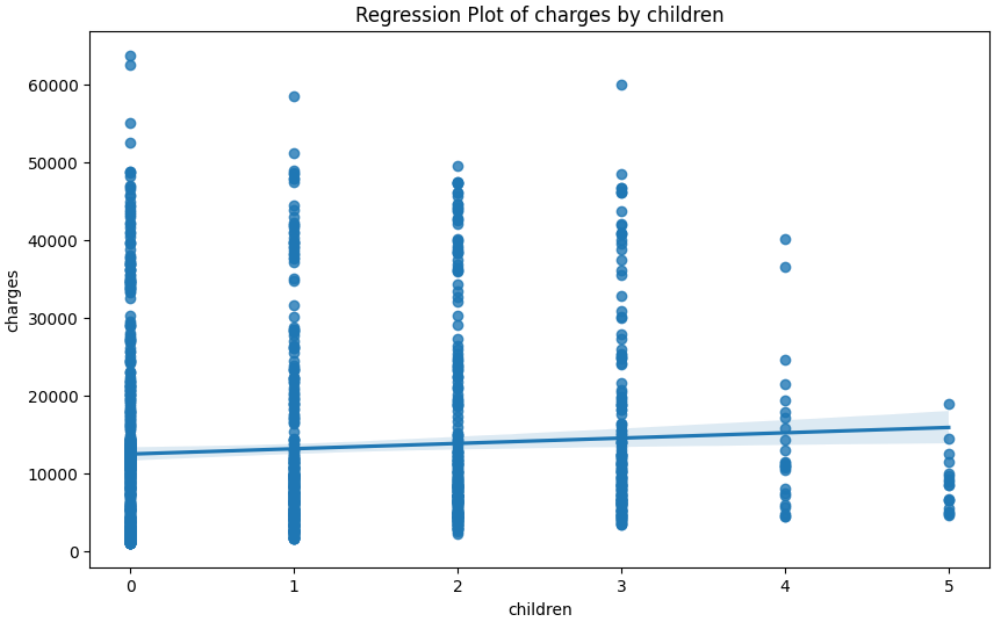


In [575]:
```
pearson_coef_bmi, p_value_bmi = stats.pearsonr(insurancedata['bmi'], insurancedata['charges'])
print("The Pearson Correlation Coefficient between bmi and charges is", pearson_coef_bmi, "with a P-value of P=", p_value_bmi)
```

```
The Pearson Correlation Coefficient between bmi and charges is 0.19834096883362878 with a P-value of P= 2.459085535116766e-13
```

In [576]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x='children', y='charges', data=insurancedata)
plt.title('Regression Plot of charges by children')
```

Out[576]: Text(0.5, 1.0, 'Regression Plot of charges by children')

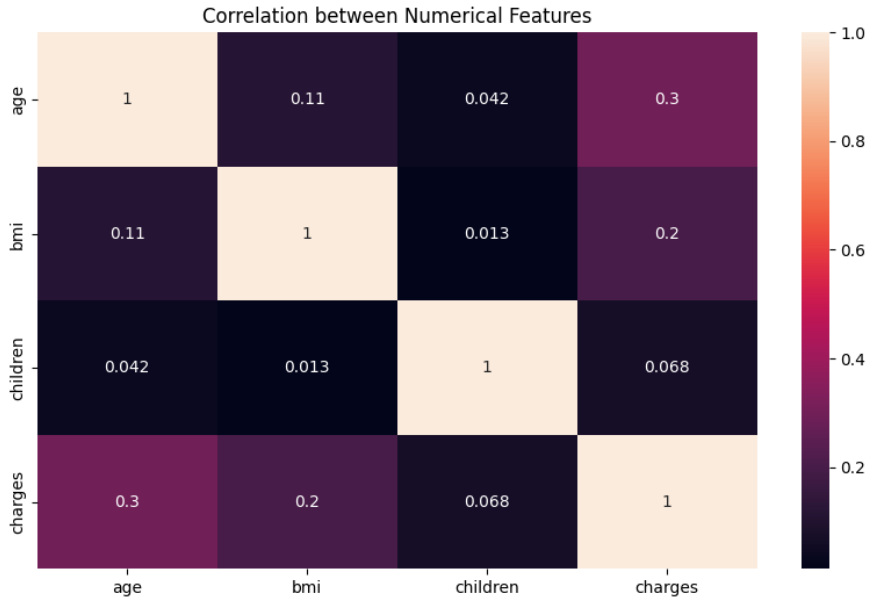


In [577]:
```
pearson_coef_children, p_value_children = stats.pearsonr(insurancedata['children'], insurancedata['charges'])
print("The Pearson Correlation Coefficient between children and charges is", pearson_coef_children, "with a P-value of P=", p_value_children)
```

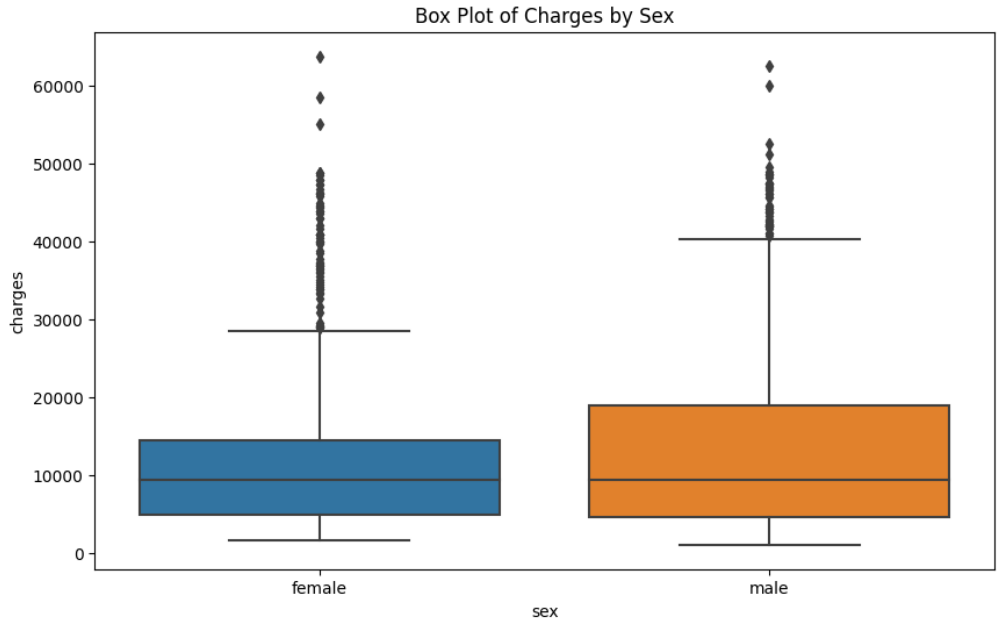
```
The Pearson Correlation Coefficient between children and charges is 0.06799822684790464 with a P-value of P= 0.012852128520136508
```

In [578]:
```
# Check the correlation between numerical features
corr_matrix = insurancedata[['age', 'bmi', 'children', 'charges']].corr()
```

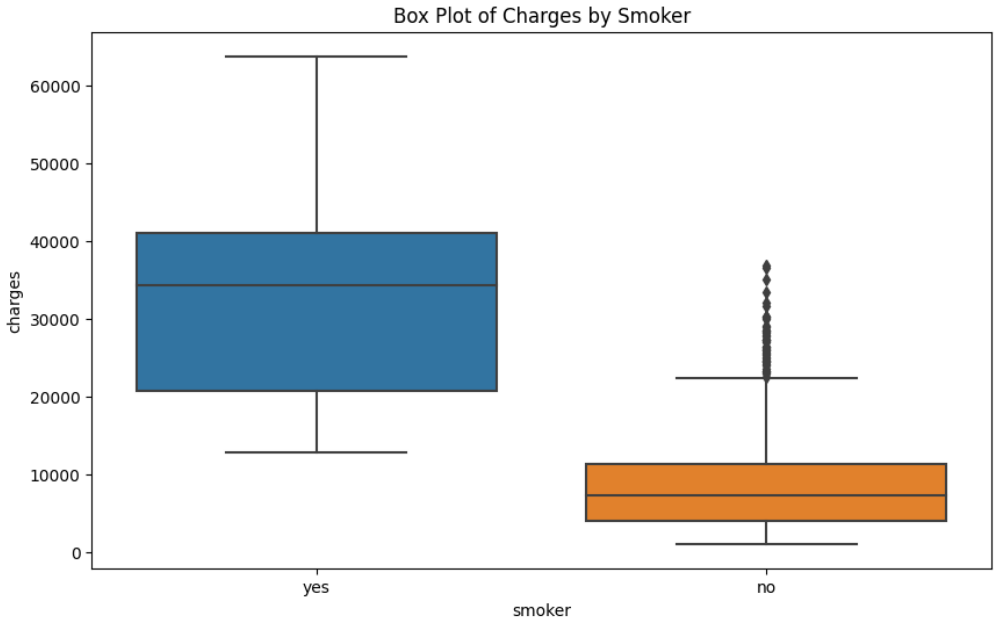
In [615]:
```
# Create a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True)
plt.title('Correlation between Numerical Features')
plt.show()
```



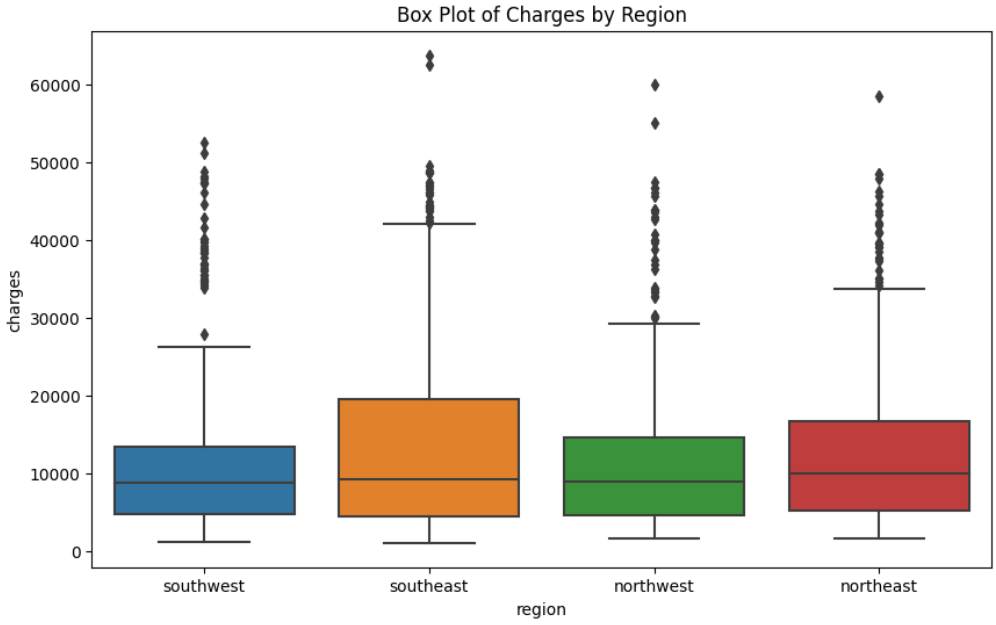
In [580]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='sex', y='charges', data=insurancedata)
plt.title('Box Plot of Charges by Sex')
plt.show()
```



In [581]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='smoker', y='charges', data=insurancedata)
plt.title('Box Plot of Charges by Smoker')
plt.show()
```



In [582]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='region', y='charges', data=insurancedata)
plt.title('Box Plot of Charges by Region')
plt.show()
```



In [583]:
```
insurancedata.drop(['region', 'sex'], axis=1, inplace = True)
```

In [584]:
```
insurancedata.shape
```

Out[584]: (1338, 5)

In [585]:
```
insurancedata.describe()
```

Out[585]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [586]:
```
insurancedata['charges']
```

Out[586]:
```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

In [587]:
```
insurancedata.head(5)
```

Out[587]:

	age	bmi	children	smoker	charges
0	19	27.900	0	yes	16884.92400
1	18	33.770	1	no	1725.55230
2	28	33.000	3	no	4449.46200
3	33	22.705	0	no	21984.47061
4	32	28.880	0	no	3866.85520

In [588]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [589]:
```
insurancedata.describe()
```

Out[589]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [590]:
```
insurancedata.describe(include=['object'])
```

Out[590]:

	smoker
count	1332
unique	2
top	no
freq	1060

In [591]:
```
insurancedata.shape
```

Out[591]:
```
(1338, 5)
```

In [592]:
```
# Check for missing values
print(insurancedata.isnull().sum())
```

```
age         0
bmi         0
children    0
smoker      6
charges     0
dtype: int64
```

In [593]:
```
insurancedata=insurancedata.dropna()
```

In [594]:
```
# Check for missing values
print(insurancedata.isnull().sum())
```

```
age         0
bmi         0
children    0
smoker      0
charges     0
dtype: int64
```

In [596]:
```
from sklearn.preprocessing import LabelEncoder
# Label encoding
label_encoder = LabelEncoder()
#insurancedata['sex'] = label_encoder.fit_transform(insurancedata['sex'])
insurancedata['smoker'] = label_encoder.fit_transform(insurancedata['smoker'])
#insurancedata['region'] = label_encoder.fit_transform(insurancedata['region'])
```

In [597]:
```
insurancedata.head(10)
```

Out[597]:

	age	bmi	children	smoker	charges
0	19	27.900	0	1	16884.92400
1	18	33.770	1	0	1725.55230
2	28	33.000	3	0	4449.46200
3	33	22.705	0	0	21984.47061
4	32	28.880	0	0	3866.85520
5	31	25.740	0	0	3756.62160
6	46	33.440	1	0	8240.58960
7	37	27.740	3	0	7281.50560
8	37	29.830	2	0	6406.41070
9	60	25.840	0	0	28923.13692

In [598]:
```
import scipy.stats as stats
insurancedata = stats.zscore(insurancedata)
```

In [599]:
```
insurancedata
```

Out[599]:

	age	bmi	children	smoker	charges
0	-1.442784	-0.453597	-0.910113	1.974097	0.301344
1	-1.514128	0.509898	-0.080927	-0.506561	-0.955793
2	-0.800690	0.383511	1.577447	-0.506561	-0.729905
3	-0.443971	-1.306298	-0.910113	-0.506561	0.724240
4	-0.515314	-0.292741	-0.910113	-0.506561	-0.778219
...	...	...	...	...	...
1333	0.768874	0.050309	1.577447	-0.506561	-0.219807
1334	-1.514128	0.206241	-0.910113	-0.506561	-0.915952
1335	-1.514128	1.015446	-0.910113	-0.506561	-0.963731
1336	-1.300096	-0.798288	-0.910113	-0.506561	-0.932375
1337	1.553656	-0.261554	-0.910113	1.974097	1.317747

1332 rows × 5 columns

In [600]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import seaborn as sns
import matplotlib.pyplot as plt
```

In [601]:
```
X = insurancedata.drop(['charges'], axis=1)
y = insurancedata['charges']
```

In [602]:
```
# Split the data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [603]:
```
# Multiple Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[603]:
```
LinearRegression()
```

In [604]:
```
# Making predictions
y_pred_lr = lr.predict(X_test)
```

In [605]:
```
# Evaluate the model
mse_lr = mean_squared_error(y_test, y_pred_lr)
print('MSE for Linear Regression: ', mse_lr)
```

```
MSE for Linear Regression:  0.2982946309844949
```

In [606]:
```
# Random Forest Regressor
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
```

Out[606]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [607]:
```
# Making predictions
y_pred_rf = rf.predict(X_test)
```

In [608]:
```
# Evaluate the model
mse_rf = mean_squared_error(y_test, y_pred_rf)
print('MSE for Random Forest: ', mse_rf)
```

```
MSE for Random Forest:  0.21307354513868598
```

In [609]:
```
# LASSO Regression
lasso = Lasso()
lasso.fit(X_train, y_train)
```

Out[609]:
```
Lasso
Lasso()
```

In [610]:
```
# Making predictions
y_pred_lasso = lasso.predict(X_test)
```

In [611]:
```
# Evaluate the model
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
print('MSE for LASSO is: ', mse_lasso)
```

```
MSE for LASSO is:  1.0072847479154785
```

In [612]:
```
# Compare performance of models using bar plot
mse_scores = [('Linear Regression', mse_lr), ('Random Forest', mse_rf), ('LASSO', mse_lasso)]
mse_df = pd.DataFrame(data = mse_scores, columns=['Model', 'MSE Score'])
mse_df.sort_values(by='MSE Score', ascending=True, inplace=True)
```

In [613]:
```
f, axe = plt.subplots(1,1, figsize=(10,5))
sns.barplot(x = mse_df['Model'], y = mse_df['MSE Score'], ax = axe)
plt.title('MSE Comparison')
plt.xlabel('Model')
plt.ylabel('MSE')
plt.show()
```