Predict Miles per Gallon (MPG) of cars and explain which features have the highest impact on the prediction result

In [ ]:
```
# Load libraries
```

In [71]:
```
import xgboost
import shap
#from __future__ import absolute_import, division, print_function, unicode_literals
import pathlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import pandas as pd
print(tf.__version__)
```

```
1.14.0
```

In [72]:
```
# load JS visualization code to notebook (will display a small js symbol inline)
shap.initjs()
```

In [ ]:

In [ ]:

Load data and display

In [76]:
```
h = keras.utils.get_file("auto-mpg.data", "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/aut
h
```

Out[76]:
```
'/Users/danielmueller/.keras/datasets/auto-mpg.data'
```

In [81]:
```
# raw_dataset.head
column_names = ['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
raw_dataset = pd.read_csv(dataset_path, names=column_names,
                      na_values = "?", comment='\t',
                      sep=" ", skipinitialspace=True)

dataset = raw_dataset.copy()
dataset.tail()
```

Out[81]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

In [86]:
```
# check of zeros
dataset.isna().sum()
```

Out[86]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

In [87]:
```
dataset.shape
```

Out[87]:
```
(398, 8)
```

In [88]:
```
# Remove lines with zeros (Alternative: impute)
dataset = dataset.dropna()
```

In [89]:
```
# check if those 6 lines have been removed
dataset.shape
```

Out[89]:
```
(392, 8)
```

Hot encoding of categorical variables

In [93]:
```
# check for unique categorical variables in columns to be encoded
dataset['Origin'].unique()
```

Out[93]:
```
array([1, 3, 2])
```

In [94]:
```
dataset['Origin'] = dataset['Origin'].map(lambda x: {1: 'USA', 2: 'Europe', 3: 'Japan'}.get(x))
```

In [95]:
```
dataset = pd.get_dummies(dataset, prefix='', prefix_sep='')
dataset.tail()
```

Out[95]:

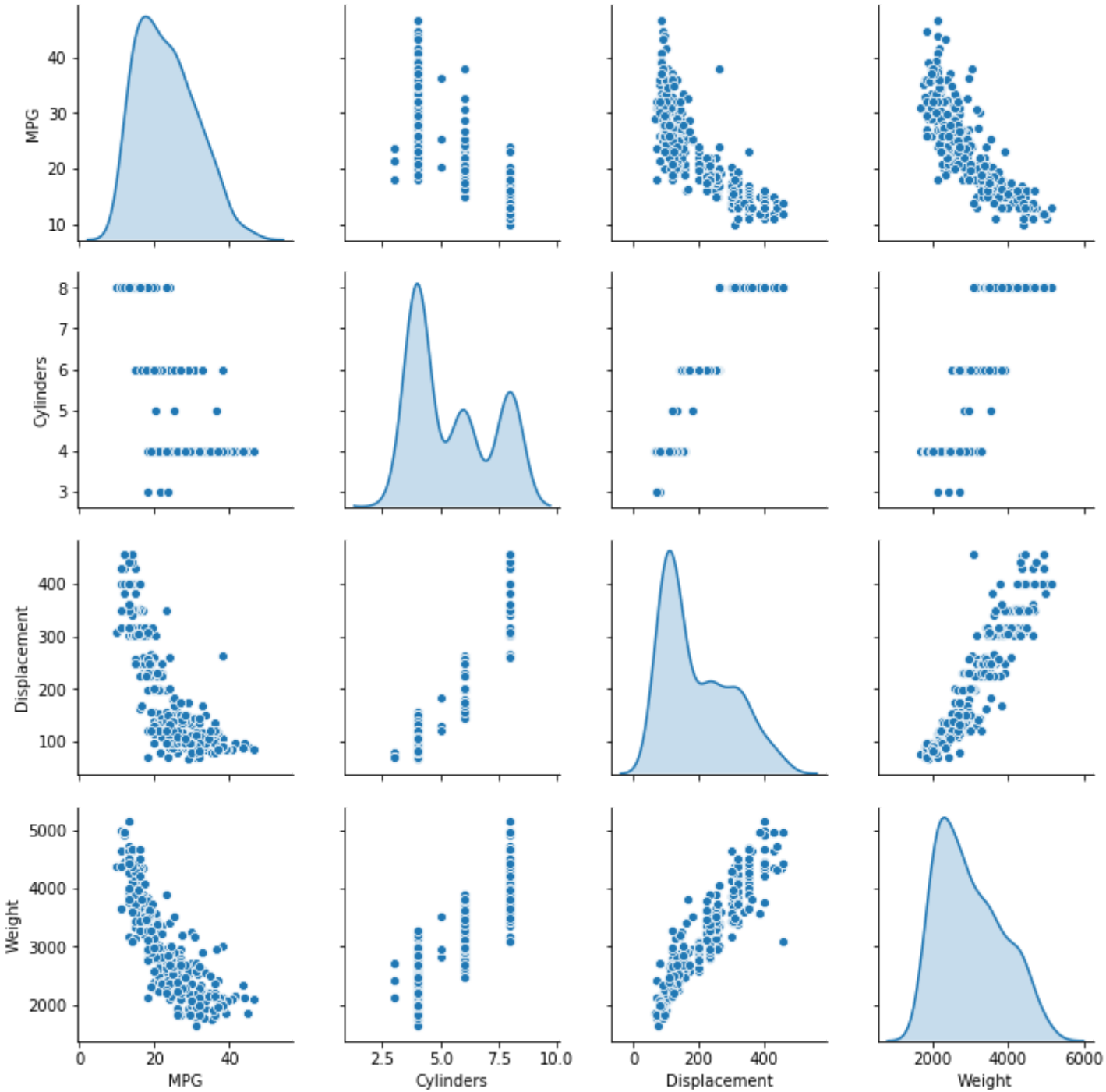
	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
393	27.0	4	140.0	86.0	2790.0	15.6	82	0	0	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	1	0	0
395	32.0	4	135.0	84.0	2295.0	11.6	82	0	0	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	0	0	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	0	0	1

In [97]:
```
train_dataset = dataset.sample(frac=0.8,random_state=0)
test_dataset = dataset.drop(train_dataset.index)
```

Visualize main features against each other

In [98]:
```
sns.pairplot(train_dataset[["MPG", "Cylinders", "Displacement", "Weight"]], diag_kind="kde")
```

Out[98]:
```
<seaborn.axisgrid.PairGrid at 0x1a43fddfd0>
```


MPG
Cylinders
Displacement
Weight
MPG
Cylinders
Displacement
Weight

Create labels by poping target column

In [100]:
```
train_labels = train_dataset.pop('MPG')
test_labels = test_dataset.pop('MPG')
```

In [101]:
```
# Redefine model input
```

In [102]:
```
X = train_dataset
y = train_labels
```

In [110]:
```
X.describe()
```

Out[110]:

	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
count	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000
mean	5.477707	195.318471	104.869427	2990.251592	15.559236	75.898089	0.178344	0.197452	0.624204
std	1.699788	104.331589	38.096214	843.898596	2.789230	3.675642	0.383413	0.398712	0.485101
min	3.000000	68.000000	46.000000	1649.000000	8.000000	70.000000	0.000000	0.000000	0.000000
25%	4.000000	105.500000	76.250000	2256.500000	13.800000	73.000000	0.000000	0.000000	0.000000
50%	4.000000	151.000000	94.500000	2822.500000	15.500000	76.000000	0.000000	0.000000	1.000000
75%	8.000000	265.750000	128.000000	3608.000000	17.200000	79.000000	0.000000	0.000000	1.000000
max	8.000000	455.000000	225.000000	5140.000000	24.800000	82.000000	1.000000	1.000000	1.000000

In [ ]:
```
```

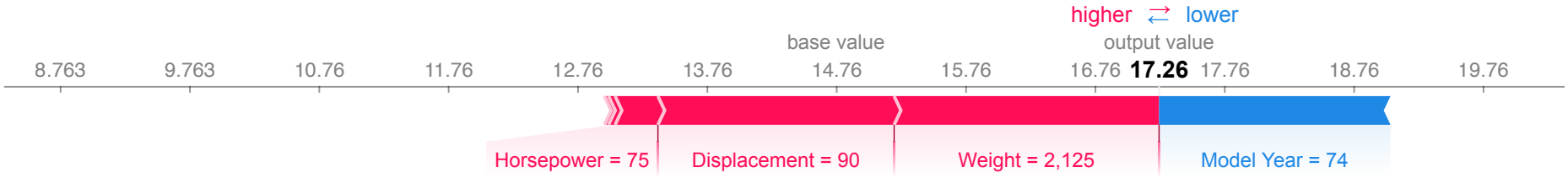
In [111]:
```
# train XGBoost model with X as features and y as label
model = xgboost.train({"learning_rate": 0.01}, xgboost.DMatrix(X, label=y), 100)
```

```
/Users/danielmueller/anaconda3/lib/python3.7/site-packages/xgboost/core.py:587: FutureWarning: Series.base is
deprecated and will be removed in a future version
  if getattr(data, 'base', None) is not None and \
/Users/danielmueller/anaconda3/lib/python3.7/site-packages/xgboost/core.py:588: FutureWarning: Series.base is
deprecated and will be removed in a future version
  data.base is not None and isinstance(data, np.ndarray) \
```

In [112]:
```
# explain the model's predictions using SHAP values
# (same syntax works for LightGBM, CatBoost, scikit-learn and spark models)
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
```

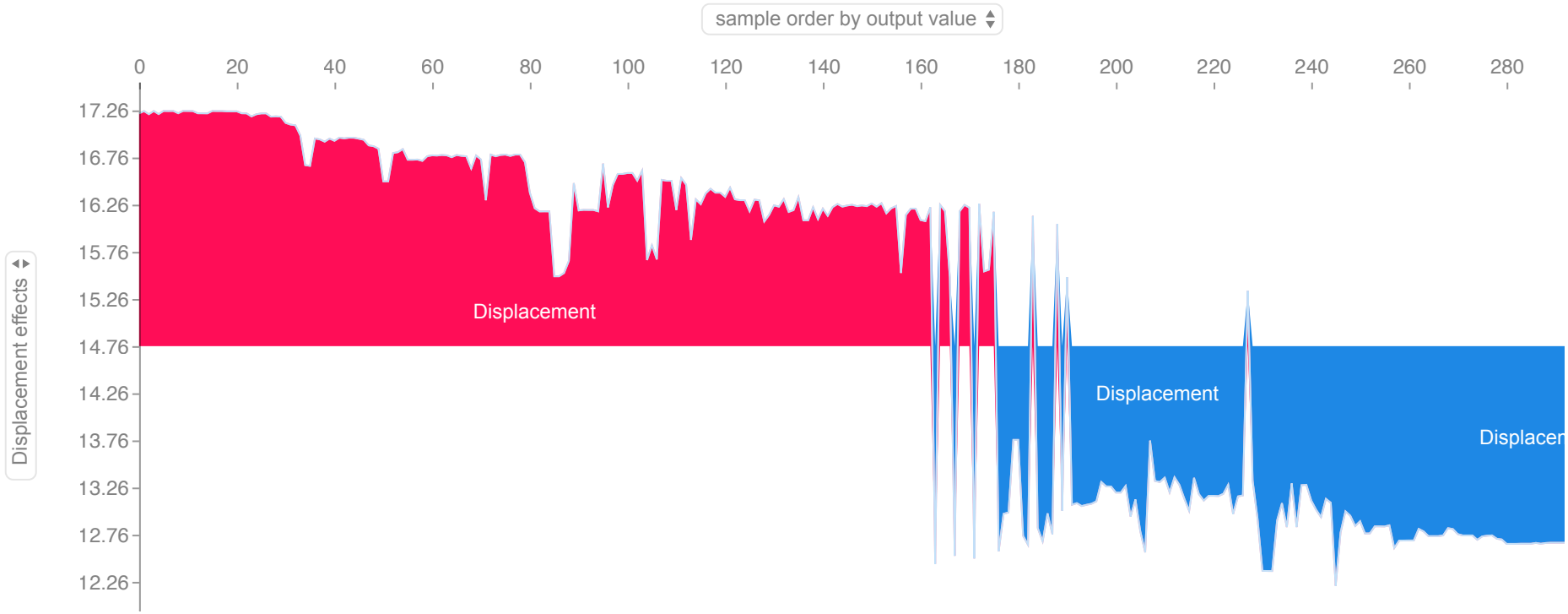
In [113]:
```
# visualize the first prediction's explanation (use matplotlib=True to avoid Javascript)
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```

Out[113]:

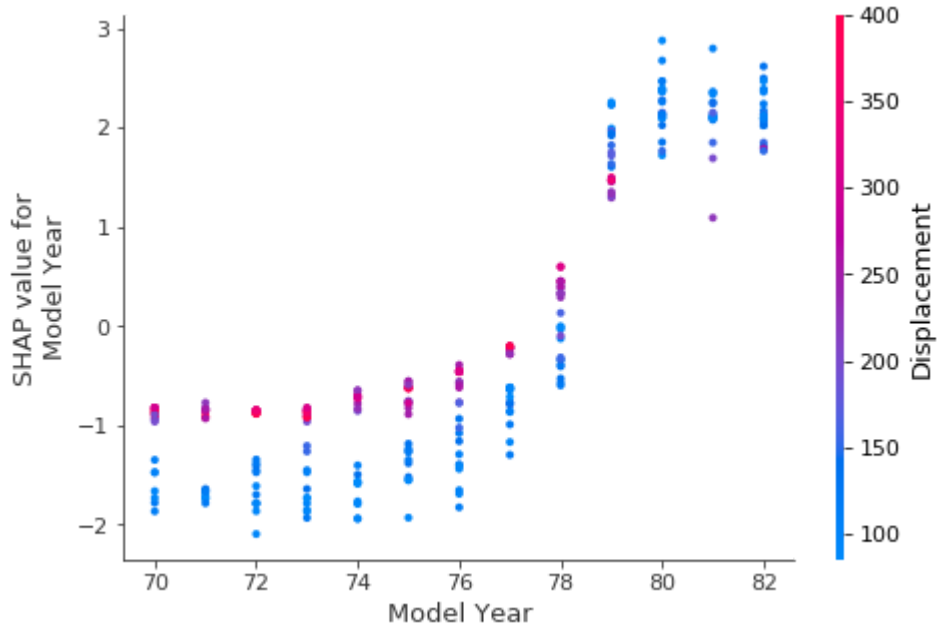


In [114]:
```
# visualize the training set predictions
shap.force_plot(explainer.expected_value, shap_values, X)
```

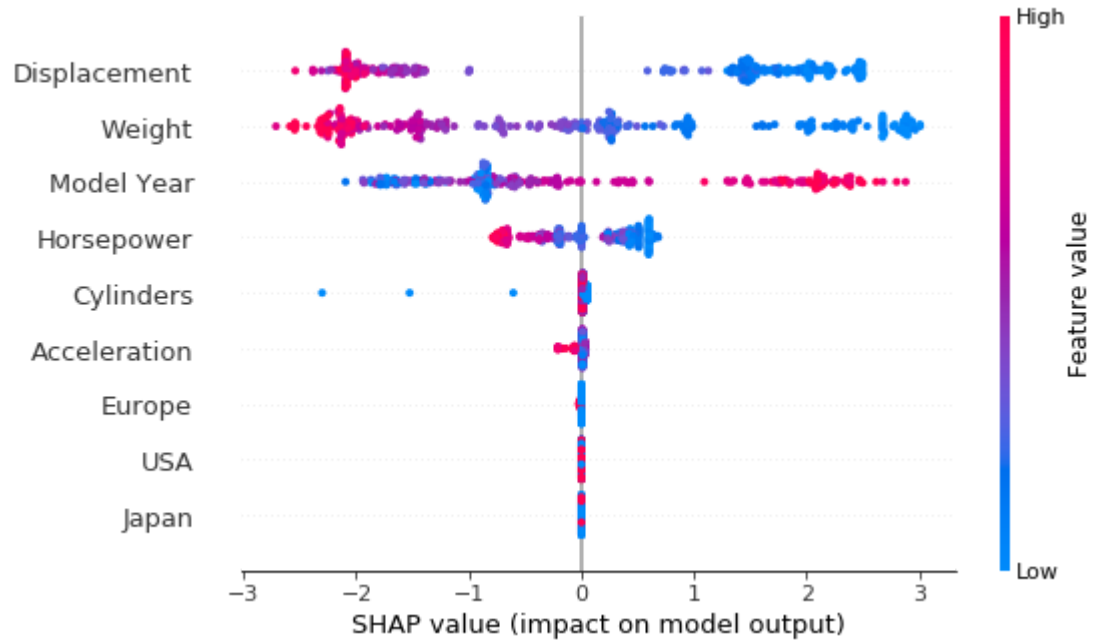
Out[114]:



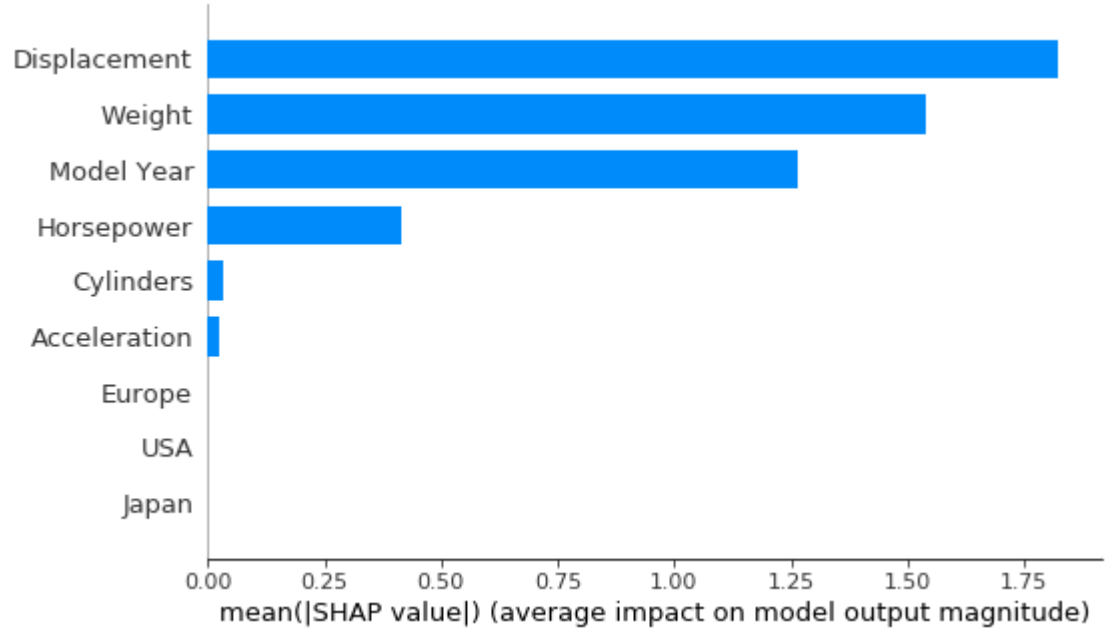
In [118]:
```
# create a SHAP dependence plot to show the effect of a single feature across the whole dataset
#shap.dependence_plot("Weight", shap_values, X)
shap.dependence_plot("Model Year", shap_values, X)
```



In [119]:
```
# summarize the effects of all the features
shap.summary_plot(shap_values, X)
```



In [68]:
```
shap.summary_plot(shap_values, X, plot_type="bar")
```



In [ ]:
```

```