In [2]:
```
import numpy as np
import pandas as pd
from functools import reduce
```

In [3]:
```
df_index=pd.read_excel('data/CSUSHPINSA wb.xlsx')
df_market=pd.read_excel('data/NASDAQ100 wb.xlsx')
df_pop=pd.read_excel('data/POPTHM wb.xlsx')
df_construction=pd.read_excel('data/WPUSI012011 wb.xlsx')
df_unemp=pd.read_excel('data/UNRATE wb.xlsx')
df_interest=pd.read_excel('data/BOGZ1FL075035503Q wb.xlsx')
df_mortgage=pd.read_excel('data/MORTGAGE30US wb.xlsx')
df_percapinc=pd.read_excel('data/percapinc.xls')
df_hunits=pd.read_excel('data/houseunits.xls')
```

In [4]:
```
data=[df_market,df_pop,df_construction,df_unemp,df_interest,df_mortgage,df_percapinc,df_
```

In [5]:
```
for dataset in data:
    print(dataset.shape)
```

```
(90, 2)
(90, 2)
(90, 2)
(90, 2)
(90, 2)
(90, 2)
(90, 2)
(90, 2)
```

In [6]:
```
for dataset in data:
    print(dataset.columns)
```

```
Index(['Quarter', 'NASDAQ100'], dtype='object')
Index(['Quarter', 'POPTHM'], dtype='object')
Index(['Quarter', 'WPUSI012011'], dtype='object')
Index(['Quarter', 'UNRATE'], dtype='object')
Index(['Quarter', 'Interest_rate'], dtype='object')
Index(['Quarter', 'MORTGAGE30US'], dtype='object')
Index(['Quarter', 'percapinc'], dtype='object')
Index(['Quarter', 'housunits'], dtype='object')
```

In [ ]:
```
```

In [7]:
```
dfs = [df_index,df_market, df_interest, df_construction,df_mortgage,df_pop,df_unemp,df_p

#merge all DataFrames into one
df = reduce(lambda  left,right: pd.merge(left,right,on=['Quarter'],
                                         how='outer'), dfs)
```

In [8]:
```
df.head()
```

Out[8]:

	Quarter	CSUSHPINSA	NASDAQ100	Interest_rate	WPUSI012011	MORTGAGE30US	POPTHM	UNRA
0	2000 Q1	100.679000	4046.825397	127696	144.733333	8.256923	281304.333333	4.033
1	2000 Q2	103.698667	3629.497460	126924	145.166667	8.316154	282002.000000	3.933
2	2000 Q3	106.459000	3779.890000	139947	143.833333	8.020000	282768.666667	4.000
3	2000 Q4	108.270000	2941.222857	144433	142.833333	7.620769	283518.666667	3.900
4	2001 Q1	109.749333	2168.620952	143811	142.266667	7.006923	284168.666667	4.233

In [37]:
```
df.rename(columns={'CSUSHPINSA':'Index','WPUSI012011':'cons_mat'},inplace=True)
```

In [38]:
```
df.shape
```

Out[38]:
```
(90, 10)
```

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 90 entries, 0 to 89
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Quarter        90 non-null     object 
 1   Index          90 non-null     float64
 2   NASDAQ100      90 non-null     float64
 3   Interest_rate  90 non-null     int64  
 4   cons_mat       90 non-null     float64
 5   MORTGAGE30US   90 non-null     float64
 6   POPTHM         90 non-null     float64
 7   UNRATE         90 non-null     float64
 8   percapinc      90 non-null     int64  
 9   housunits      90 non-null     int64  
dtypes: float64(6), int64(3), object(1)
memory usage: 7.7+ KB
```

In [40]:
```
df.describe(include="all")
```

Out[40]:

	Quarter	Index	NASDAQ100	Interest_rate	cons_mat	MORTGAGE30US	POPTHM	U
count	90	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000	90
unique	90	NaN	NaN	NaN	NaN	NaN	NaN	
top	2000 Q1	NaN	NaN	NaN	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	169.041648	4100.429369	216455.144444	200.884230	4.985707	310426.674074	5
std	NaN	41.061039	3610.055121	58960.683183	43.961087	1.359509	16100.877054	1
min	NaN	100.679000	945.332813	126924.000000	142.033333	2.760714	281304.333333	3
25%	NaN	141.962583	1633.297222	169301.500000	170.400000	3.887349	296526.666667	4
50%	NaN	165.653000	2588.831357	212705.500000	201.950000	4.717692	311658.333333	5
75%	NaN	184.360417	4814.456295	254024.000000	215.683333	6.084038	325399.333333	6
max	NaN	305.348000	15843.419219	360001.000000	348.737333	8.316154	332939.666667	12

In [42]:
```
#Finding correlation
df.corr()['Index'].sort_values(ascending=False)
```

Out[42]:
```
Index           1.000000
Interest_rate   0.956532
cons_mat        0.896744
percapinc       0.877490
NASDAQ100       0.849974
housunits       0.814834
POPTHM          0.771812
UNRATE         -0.262429
MORTGAGE30US   -0.583233
Name: Index, dtype: float64
```

In [43]:
```
#No missing values
df.isnull().sum()
```

Out[43]:
```
Quarter         0
Index           0
NASDAQ100       0
Interest_rate   0
cons_mat        0
MORTGAGE30US    0
POPTHM          0
UNRATE          0
percapinc       0
housunits       0
dtype: int64
```

In [44]:
```
df.to_excel('data/data.xlsx',index=False)
```

In [46]:
```
x=df.drop(["Index","Quarter"],axis=True)
y=df["Index"]
```

In [56]:
```
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
```

In [48]:
```
#training random forest model
reg = RandomForestRegressor(n_estimators=100)
reg.fit(x,y)
```

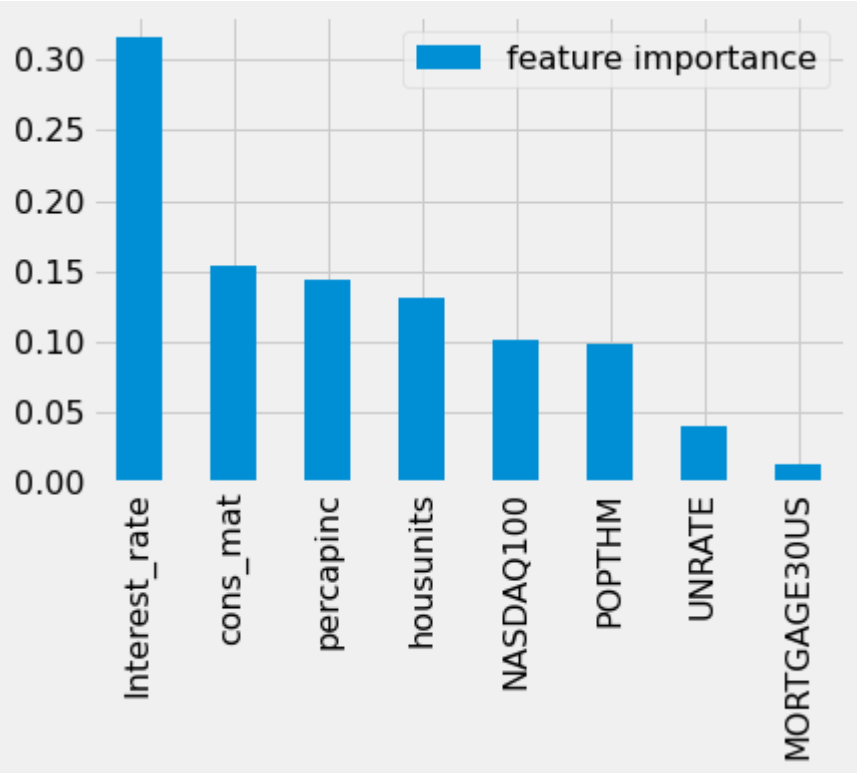
Out[48]:
```
RandomForestRegressor()
```

In [49]:
```
#feature importance
df_feature_importance = pd.DataFrame(reg.feature_importances_, index=x.columns, columns=
df_feature_importance
```

Out[49]:

	feature importance
Interest_rate	0.315570
cons_mat	0.153915
percapinc	0.144698
housunits	0.131795
NASDAQ100	0.101607
POPTHM	0.098465
UNRATE	0.040440
MORTGAGE30US	0.013510

In [50]:
```
df_feature_importance.plot(kind='bar');
```



In [51]:
```
#training xgboost regressor
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
```

In [52]:
```
xgbregressor=xgb.XGBRegressor(eval_metric='rmsle')
```

In [53]:
```
param_grid = {"max_depth":    [4, 5],
              "n_estimators": [500, 600, 700],
              "learning_rate": [0.01, 0.015]}

# try out every combination of the above values
search = GridSearchCV(xgbregressor, param_grid, cv=5).fit(x,y)

print("The best hyperparameters are ",search.best_params_)
```

```
The best hyperparameters are  {'learning_rate': 0.015, 'max_depth': 4, 'n_estimators':
600}
```

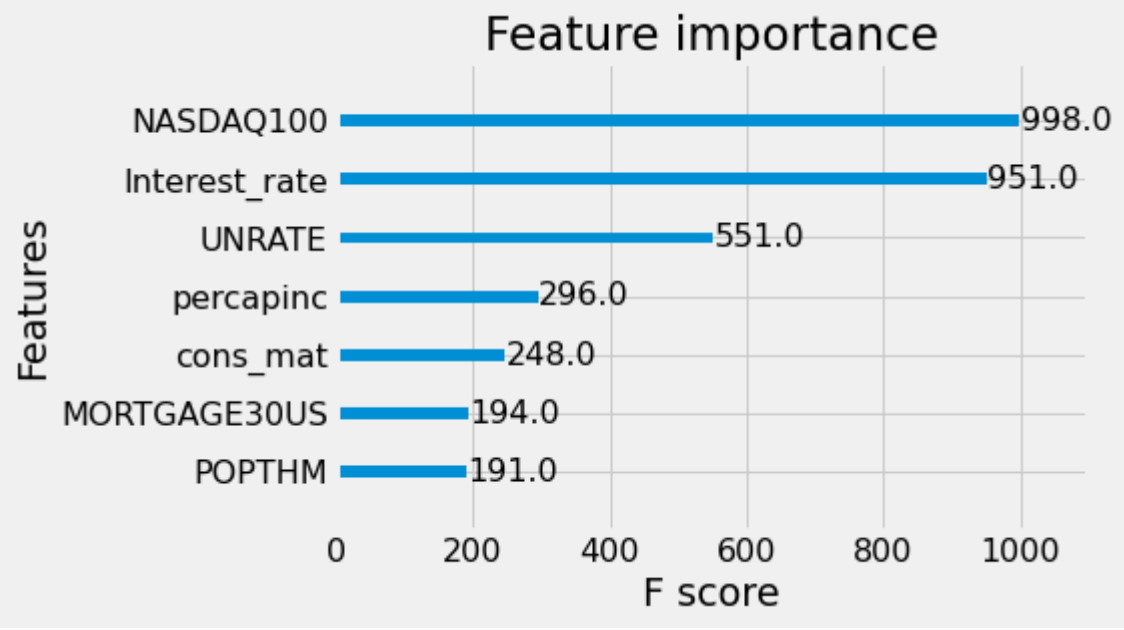
In [54]:
```
xgbregressor=xgb.XGBRegressor(learning_rate = search.best_params_["learning_rate"],
                              n_estimators  = search.best_params_["n_estimators"],
                              max_depth     = search.best_params_["max_depth"],
                              eval_metric='rmsle')

xgbregressor.fit(x,y)
```

Out[54]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric='rmsle', gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.015, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=4, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=600, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In [55]:
```
from xgboost import plot_importance
import matplotlib.pyplot as plt


plot_importance(xgbregressor)
plt.show()
```



In [ ]: