In [3]:

```
#Pearson's Correlation
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

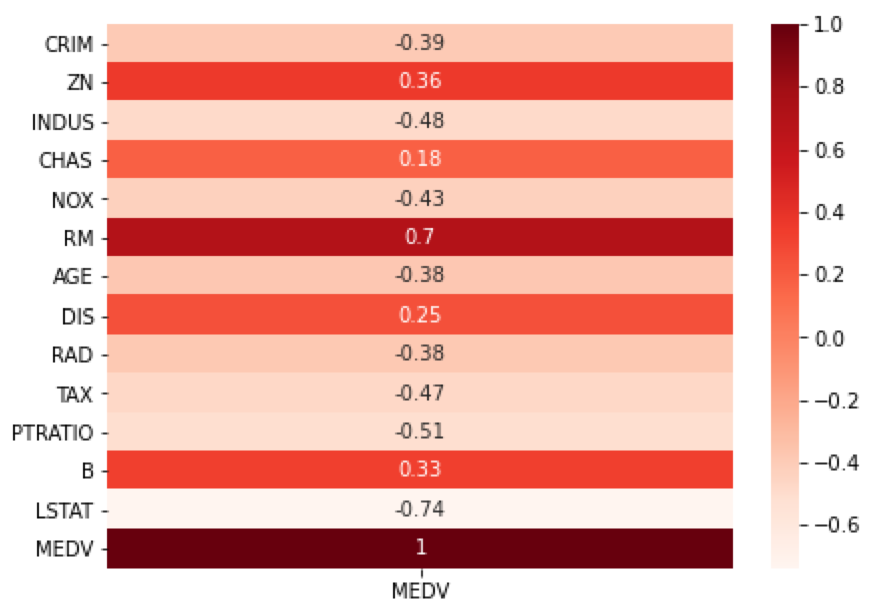
In [4]:

```
from sklearn.datasets import load_boston
X, y = load_boston(return_X_y=True)
feature_names = load_boston().feature_names
data = pd.DataFrame(X, columns=feature_names)
data['MEDV'] = y
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecated in
1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
```

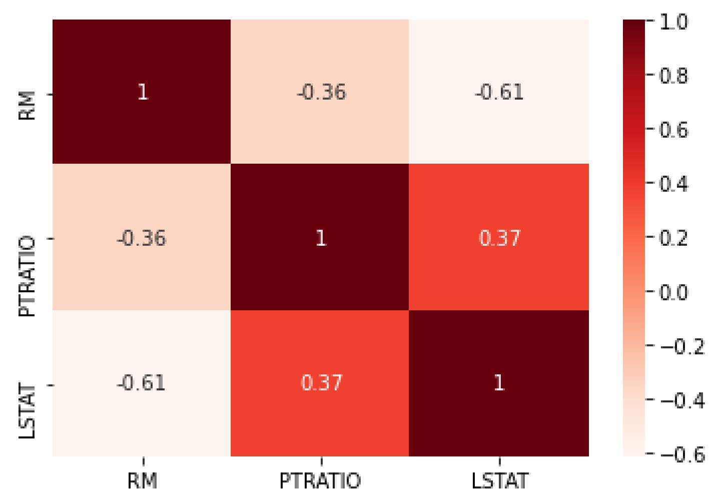
In [5]:

```
# compute pearson's r
target_correlation = data.corr()[['MEDV']]
# we only care about the target variable
plt.figure(figsize=(7,5))
sns.heatmap(target_correlation, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [6]:

```
sns.heatmap(data.corr().loc[['RM', 'PTRATIO', 'LSTAT'], ['RM', 'PTRATIO', 'LSTAT']], annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [7]:

```
#LDA
#Linear Discriminant Analysis
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.pipeline import Pipeline
```

In [8]:

```
df = pd.read_csv('cancer.csv').iloc[:,1:-1]
X = df.drop(['diagnosis'], axis=1)
le = LabelEncoder()
y = le.fit_transform(df.diagnosis)
labels = le.classes_
```

In [9]:

```
steps = [('lda', LinearDiscriminantAnalysis()), ('m', LogisticRegression(C=10))]
model = Pipeline(steps=steps)
```

In [10]:

```
# evaluate model
cv = StratifiedKFold(n_splits=5)
n_scores_lda = cross_val_score(model, X, y, scoring='f1_macro', cv=cv, n_jobs=-1)
model = LogisticRegression(C=10)
n_scores = cross_val_score(model, X, y, scoring='f1_macro', cv=cv, n_jobs=-1)
```

In [11]:

```
# report performance
print('f1-score (macro)\n')
print('With LDA: %.2f' % np.mean(n_scores_lda))
print('Without LDA: %.2f' % np.mean(n_scores))
```

```
f1-score (macro)

With LDA: 0.97
Without LDA: 0.93
```

In [12]:

```
# ANOVA
from sklearn.feature_selection import f_classif, SelectKBest
fs = SelectKBest(score_func=f_classif, k=5)

X_new = fs.fit(X, y)
```

In [19]:

```
from sklearn.model_selection import StratifiedKFold, GridSearch
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
cv = StratifiedKFold(n_splits=5)
pipeline = Pipeline(steps=[('anova',fs), ('lr', LinearRegression(solver='liblinear'))])
params = {['anova__k']: [i+1 for i in range(X.shape[1])]}
search = GridSearchCV(pipeline, params, scoring='accuracy', n_jobs=-1, cv=cv)
results = search.fit(X, y)
print('Best k: %s' % results.best_params_)
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-19-8b97af7e48d4> in <module>
----> 1 from sklearn.model_selection import StratifiedKFold, GridSearch
      2 from sklearn.pipeline import Pipeline
      3 from sklearn.linear_model import LinearRegression
      4 cv = StratifiedKFold(n_splits=5)
      5 pipeline = Pipeline(steps=[('anova',fs), ('lr', LinearRegression(solver='liblinear'))])

ImportError: cannot import name 'GridSearch' from 'sklearn.model_selection' (C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\__init__.py)
```

In [22]:

```
#χ² Chi-squared tests
from sklearn.feature_selection import chi2, SelectKBest
loan = pd.read_csv('loan_data_set.csv')
loan = loan.drop('Loan_ID', axis=1) # irrelevant feature
```

In [23]:

```
#Transform the numerical feature into categorical feature
loan['Loan_Amount_Term'] = loan['Loan_Amount_Term'].astype('object')
loan['Credit_History'] = loan['Credit_History'].astype('object')
```

In [24]:

```
#Dropping all the null value
loan.dropna(inplace = True)
```

In [25]:

```
#Retrieve all the categorical columns except the target
categorical_columns = loan.select_dtypes(exclude='number').drop('Loan_Status', axis=1).columns
X = loan[categorical_columns].apply(LabelEncoder().fit_transform)
y = LabelEncoder().fit_transform(loan['Loan_Status'])
fs = SelectKBest(score_func=chi2, k=5)
X_kbest = fs.fit_transform(X, y)
```

In [26]:

```
X_kbest
```

Out[26]:

```
array([[1, 1, 0, 0, 1],
       [1, 0, 0, 1, 1],
       [1, 0, 1, 0, 1],
       ...,
       [1, 1, 0, 0, 1],
       [1, 2, 0, 0, 1],
       [0, 0, 0, 1, 0]])
```

In [ ]: