In [1]:
```
# This Code Does an Import of a CSV file an alternative may be an excel file
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import StandardScaler
pd.options.mode.chained_assignment = None

import warnings
warnings.filterwarnings('ignore')


#Phase 1 collecting the data
pd.set_option("expand_frame_repr", False) #Avoids Printing on the next line
df= pd.read_csv('C:/Users/Marc/Dropbox/University of Pretoria/791/Cheat Shee
df.columns =["CRIM","ZN","INDUS","CHAS","NOX","RM","AGE","DIS","RAD","TAX","
df
```

Out[1]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRA
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	
...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	

506 rows × 13 columns

In [2]:
```
df.describe()
```

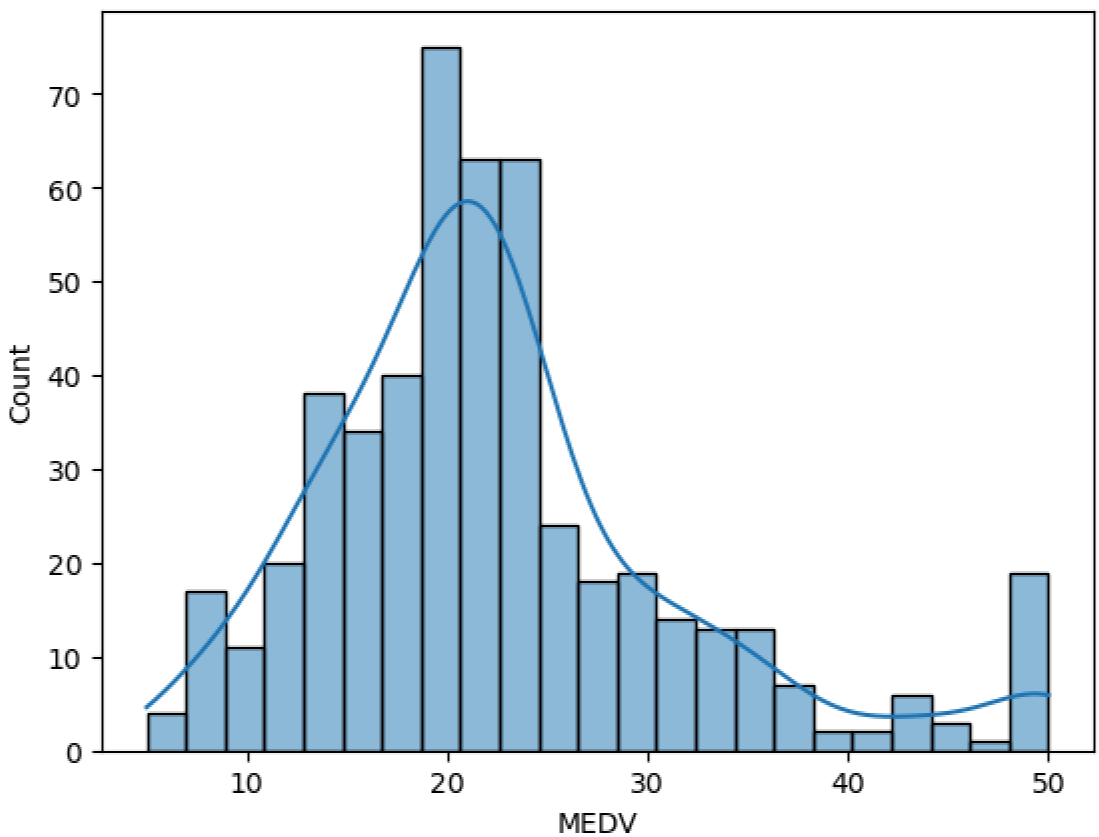
Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

In [3]:

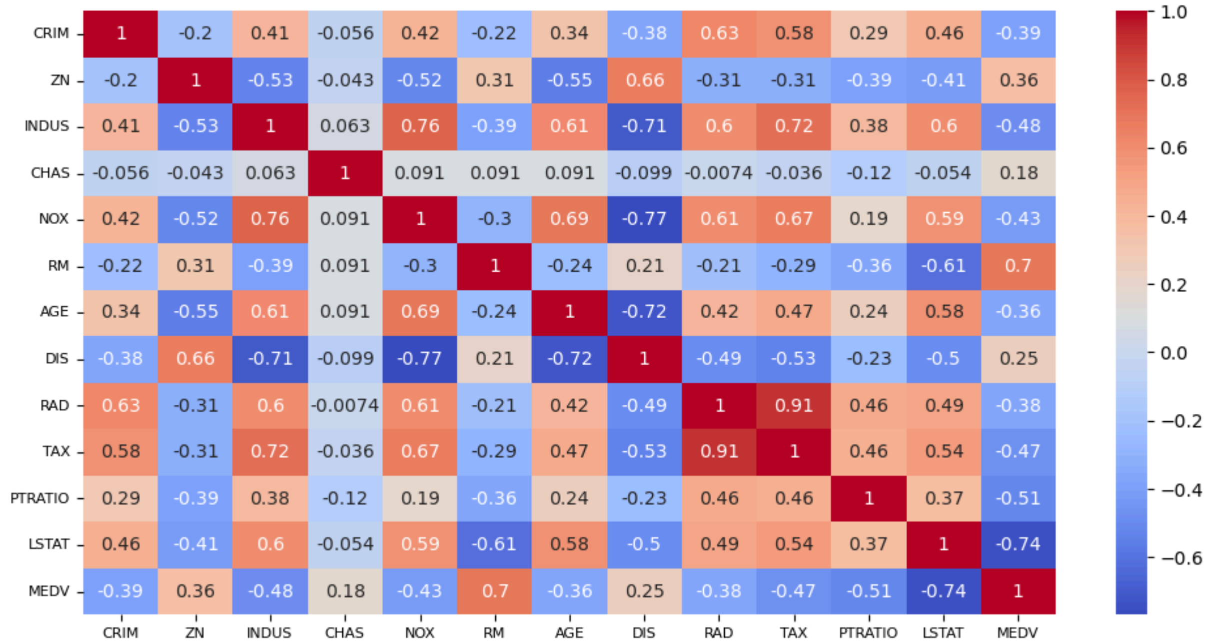
```
sns.histplot(data=df, x="MEDV", kde=True)
```

Out[3]: <Axes: xlabel='MEDV', ylabel='Count'>

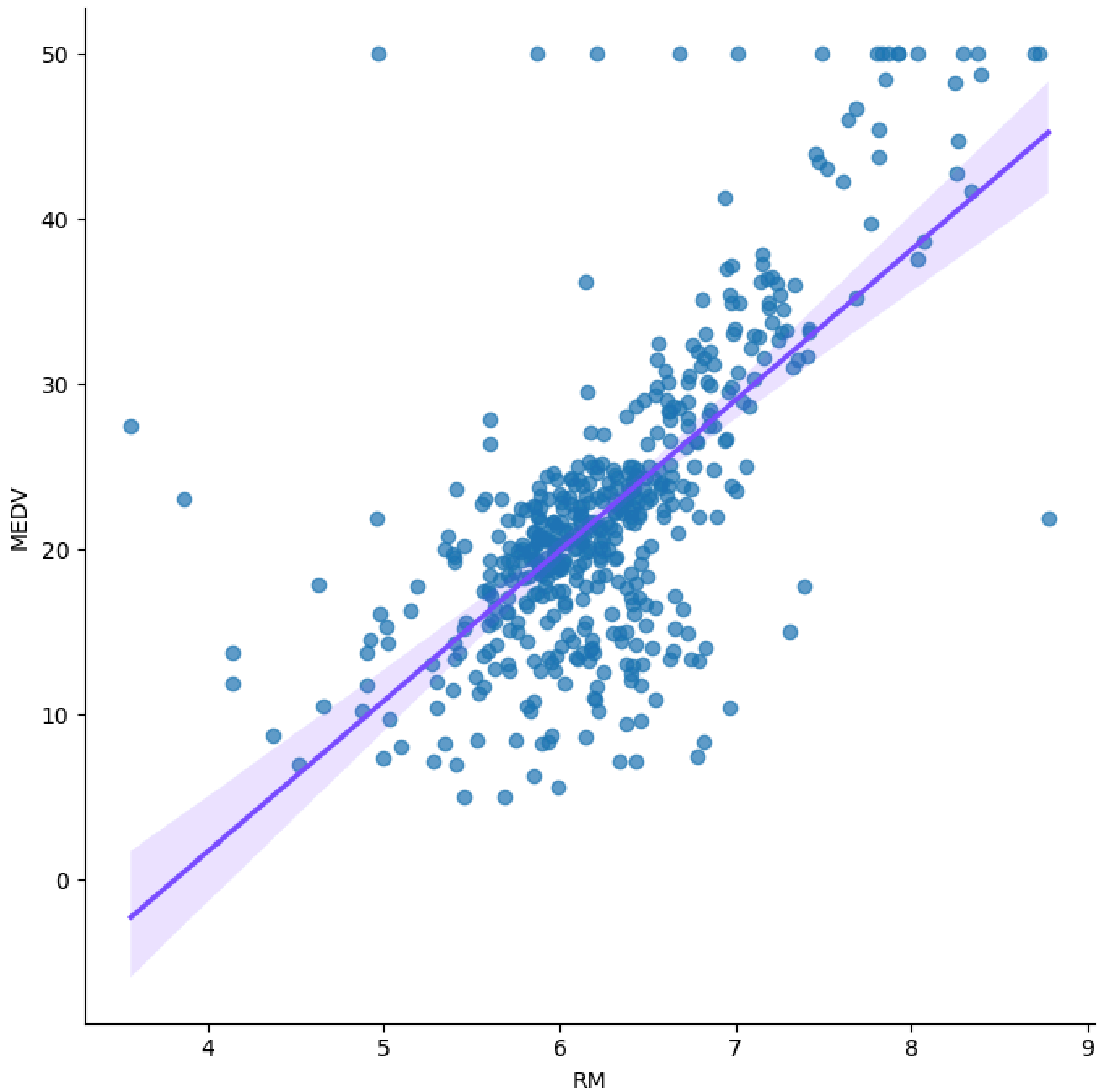


In [4]:

```
plt.figure(figsize=(12, 6))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.show()
```



In [5]:
```
sns.lmplot(x="RM", y="MEDV", data=df, height=7, scatter_kws={'alpha':0.7}, l
plt.show()
```




```
In [6]: #Can we actually determine the type of species based on the bill length, bil
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import StackingClassifier #ensmbl method of stacking c
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.linear_model import LinearRegression


from sklearn.tree import DecisionTreeClassifier   #estimator in GA
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
# Convert levels to numeric
feature_encoder= LabelEncoder()
df['CRIM'] = feature_encoder.fit_transform(df['CRIM'])
df['ZN'] = feature_encoder.fit_transform(df['ZN'])
df['INDUS'] = feature_encoder.fit_transform(df['INDUS'])
df['CHAS'] = feature_encoder.fit_transform(df['CHAS'])
df['NOX'] = feature_encoder.fit_transform(df['NOX'])
df['RM'] = feature_encoder.fit_transform(df['RM'])
df['AGE'] = feature_encoder.fit_transform(df['AGE'])
df['DIS'] = feature_encoder.fit_transform(df['DIS'])
df['RAD'] = feature_encoder.fit_transform(df['RAD'])
df['TAX'] = feature_encoder.fit_transform(df['TAX'])
df['PTRATIO'] = feature_encoder.fit_transform(df['PTRATIO'])
df['LSTAT'] = feature_encoder.fit_transform(df['LSTAT'])


# Define the input features (Defender Score, Attacker Score, Log Time)
X = df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TA
y = df['MEDV']

# Split the data into training and testing sets (80% train, 20% test)
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, rar

# Output the shapes of the training and test sets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[7]: ((404, 12), (102, 12), (404,), (102,))

In [8]:
```
regr = LinearRegression()
regr.fit(X_train, y_train)
print('Training data r-squared:', regr.score(X_train, y_train))
print('Test data r-squared:', regr.score(X_test,y_test))
print('Intercept', regr.intercept_)
pd.DataFrame(data=regr.coef_, index=X_train.columns, columns=['coef'])
```

```
Training data r-squared: 0.7337895489177254
Test data r-squared: 0.717270887159696
Intercept 43.466876857735876
```

Out[8]:

	coef
CRIM	0.007699
ZN	-0.125675
INDUS	-0.044003
CHAS	3.329627
NOX	-0.080087
RM	0.012634
AGE	0.004529
DIS	-0.019261
RAD	0.211347
TAX	-0.071936
PTRATIO	-0.115548
LSTAT	-0.047236

In [9]:
```
df['MEDV'].skew()
```

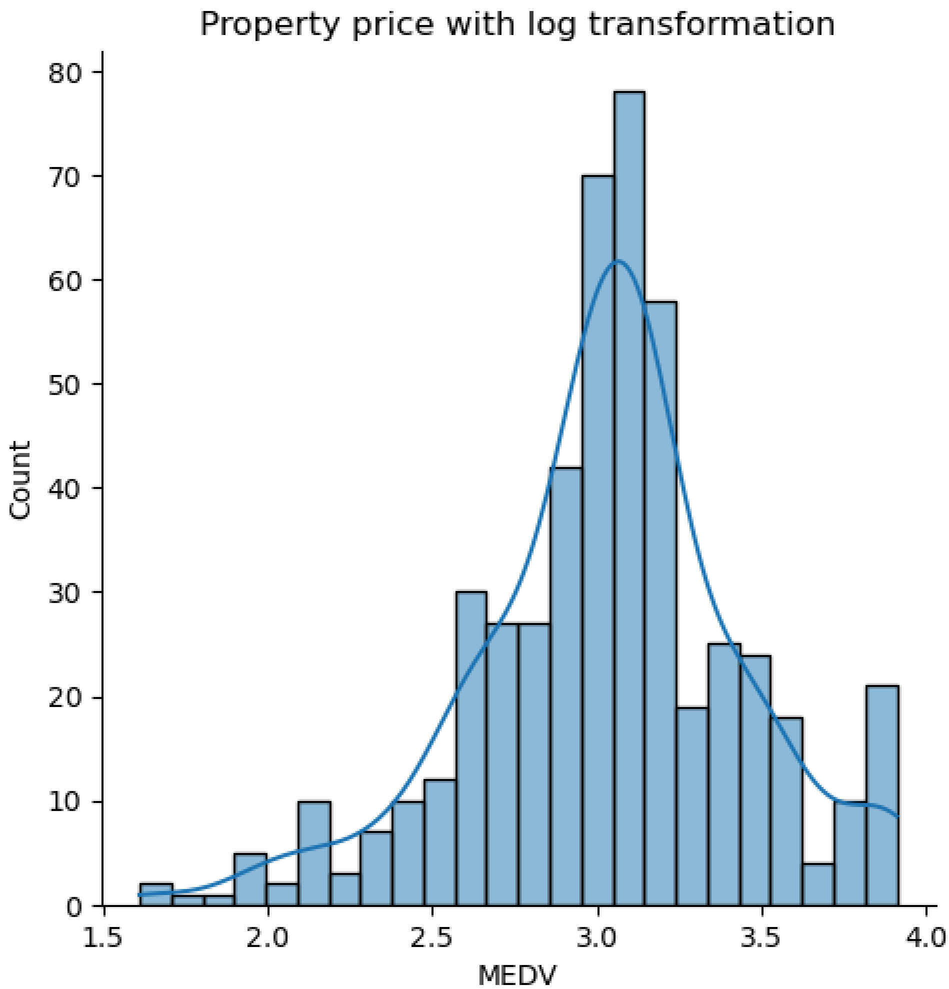
Out[9]: 1.1080984082549072

In [10]:
```
medv_log_transformed = np.log(df["MEDV"])
medv_log_transformed.skew()
```

Out[10]: -0.33032129530987864

In [11]:
```
sns.displot(medv_log_transformed, kde=True)
plt.title("Property price with log transformation")
```

Out[11]: Text(0.5, 1.0, 'Property price with log transformation')



In [12]:
```
df['medv_log_transformed']= np.log(df["MEDV"])
df.head()
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0	0	3	19	0	51	320	173	297	0	34	9	53
1	23	0	56	0	36	279	226	333	1	11	23	161
2	22	0	56	0	36	400	160	333	1	11	23	28
3	32	0	16	0	33	383	113	361	2	5	31	6
4	110	0	16	0	33	395	140	361	2	5	31	64

In [13]:
```
# Convert levels to numeric
feature_encoder= LabelEncoder()
df['CRIM'] = feature_encoder.fit_transform(df['CRIM'])
df['ZN'] = feature_encoder.fit_transform(df['ZN'])
df['INDUS'] = feature_encoder.fit_transform(df['INDUS'])
df['CHAS'] = feature_encoder.fit_transform(df['CHAS'])
df['NOX'] = feature_encoder.fit_transform(df['NOX'])
df['RM'] = feature_encoder.fit_transform(df['RM'])
df['AGE'] = feature_encoder.fit_transform(df['AGE'])
df['DIS'] = feature_encoder.fit_transform(df['DIS'])
Loading [MathJax]/extensions/Safe.js  = feature_encoder.fit_transform(df['RAD'])
```

```
df['TAX'] = feature_encoder.fit_transform(df['TAX'])
df['PTRATIO'] = feature_encoder.fit_transform(df['PTRATIO'])
df['LSTAT'] = feature_encoder.fit_transform(df['LSTAT'])


# Define the input features (Defender Score, Attacker Score, Log Time)
X = df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TA
y = df['medv_log_transformed']

# Split the data into training and testing sets (80% train, 20% test)
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, rar

# Output the shapes of the training and test sets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[13]: ((404, 12), (102, 12), (404,), (102,))

In [14]:

```
regr = LinearRegression()
regr.fit(X_train, y_train)
print('Training data r-squared:', regr.score(X_train, y_train))
print('Test data r-squared:', regr.score(X_test,y_test))
print('Intercept', regr.intercept_)
pd.DataFrame(data=regr.coef_, index=X_train.columns, columns=['coef'])
```

```
Training data r-squared: 0.7321586307806947
Test data r-squared: 0.7181931938758924
Intercept 3.7621342962886537
```

Out[14]:

	coef
CRIM	-0.000054
ZN	-0.007273
INDUS	-0.000229
CHAS	0.131028
NOX	-0.001763
RM	0.000393
AGE	0.000194
DIS	-0.000201
RAD	0.004861
TAX	-0.002400
PTRATIO	-0.005301
LSTAT	-0.002055

In [ ]: