In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import KFold
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [2]:
```
df = pd.read_csv('Hitters.csv')
df_copy = df.copy()
df_copy['League'] = df_copy['League'].replace({'N': 1.0, 'A': 0.0})
df_copy['Division'] = df_copy['Division'].replace({'W': 1.0, 'E': 0.0})
df_copy['NewLeague'] = df_copy['NewLeague'].replace({'N': 1.0, 'A': 0.0})
df_copy.head()
```

Out[2]:

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors
0	293	66	1	30	29	14	1	293	66	1	30	29	14	0.0	0.0	446	33	20
1	315	81	7	24	38	39	14	3449	835	69	321	414	375	1.0	1.0	632	43	10
2	479	130	18	66	72	76	3	1624	457	63	224	266	263	0.0	1.0	880	82	14
3	496	141	20	65	78	37	11	5628	1575	225	828	838	354	1.0	0.0	200	11	3
4	321	87	10	39	42	30	2	396	101	12	48	46	33	1.0	0.0	805	40	4

Problem a

In [3]:
```
# Remove instances with missing values
df_copy = df_copy.dropna(axis=0)
```

```
# log-transform the Salary Column
df_copy['Salary'] = np.log(df_copy['Salary'])
```

Problem b

In [5]:

```
y = df_copy['Salary']

X = df_copy.drop('Salary',1)

X_train = X.iloc[:200,:]
X_test = X.iloc[200:,:]
y_train = y.iloc[:200]
y_test = y.iloc[200:]
```

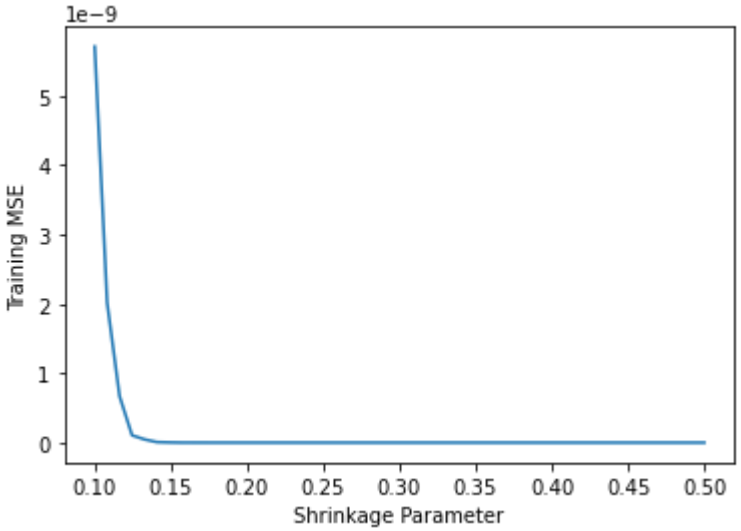
```
C:\Users\ADAMYA~1\AppData\Local\Temp/ipykernel_28264/1672455205.py:3: FutureWarning: In a future version of pandas all ar
guments of DataFrame.drop except for the argument 'labels' will be keyword-only
  X = df_copy.drop('Salary',1)
```

Problem c

In [6]:

```
lambda_list = np.linspace(0.1,0.5,50)
training_error_list = []
testing_error_list = []
for lambda_value in (lambda_list):
    # Create Gradient Boosting model and fit to training data
    GB_regressor = GradientBoostingRegressor(learning_rate=lambda_value, n_estimators=1000)
    GB_regressor.fit(X_train,y_train)

    training_y_pred = GB_regressor.predict(X_train)
    training_error = mean_squared_error(y_train,training_y_pred)

    testing_y_pred = GB_regressor.predict(X_test)
    testing_error = mean_squared_error(y_test,testing_y_pred)

    training_error_list.append(training_error)
    testing_error_list.append(testing_error)

plt.plot(lambda_list,training_error_list)
plt.xlabel("Shrinkage Parameter")
plt.ylabel("Training MSE")
```

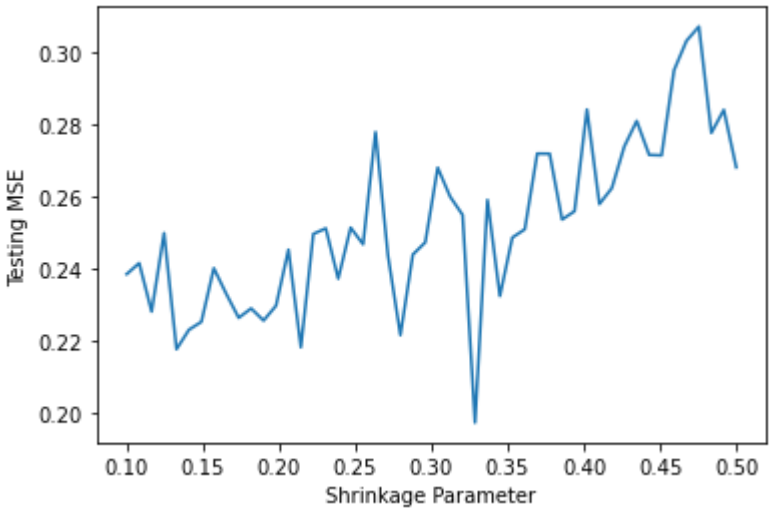
Out[6]:
```
Text(0, 0.5, 'Training MSE')
```



Problem d

In [7]:
```
plt.plot(lambda_list,testing_error_list)
plt.xlabel("Shrinkage Parameter")
plt.ylabel("Testing MSE")
```

Out[7]:
```
Text(0, 0.5, 'Testing MSE')
```



Problem e

In [66]:

```
# Simple Linear Regression
LinRegr = LinearRegression()
LinRegr.fit(X_train,y_train)

y_pred = LinRegr.predict(X_test)
MSE = mean_squared_error(y_test,y_pred)
print("The MSE for Linear Regression is " + repr(MSE))

# K-Nearest Neighbors Regression
KN_regressor = KNeighborsRegressor()
KN_regressor.fit(X_train,y_train)

y_pred = KN_regressor.predict(X_test)
MSE = mean_squared_error(y_test,y_pred)
print("The MSE for K-Neighbors Regression is " + repr(MSE))

# Gradient Boosting Regression
GB_regressor = GradientBoostingRegressor()
GB_regressor.fit(X_train,y_train)

y_pred = GB_regressor.predict(X_test)
MSE = mean_squared_error(y_test,y_pred)
print("The MSE for Gradient Boosting Regression is " + repr(MSE))
```

```
The MSE for Linear Regression is 0.49541867535174294
The MSE for K-Neighbors Regression is 0.4518306295380096
The MSE for Gradient Boosting Regression is 0.3859412214403504
```

Problem f

In [8]:

```
feature_importances = GB_regressor.feature_importances_
data = {
        'Features': list(X.columns.values),
        'Feature Importances': feature_importances
}
feat_import = pd.DataFrame(data)
feat_import
```

Out[8]:

	Features	Feature Importances
0	AtBat	0.015306
1	Hits	0.046453
2	HmRun	0.005153
3	Runs	0.001224
4	RBI	0.004384
5	Walks	0.063386
6	Years	0.020429
7	CAtBat	0.587890
8	CHits	0.031900
9	CHmRun	0.031011
10	CRuns	0.056883
11	CRBI	0.014145
12	CWalks	0.081922
13	League	0.000102
14	Division	0.000597

	Features	Feature Importances
15	PutOuts	0.019651
16	Assists	0.013429
17	Errors	0.004197
18	NewLeague	0.001938

Problem g

In [9]:
```
Bag_regressor = BaggingRegressor(base_estimator=GradientBoostingRegressor(),
                                 n_estimators=10,
                                 random_state=0)
Bag_regressor.fit(X_train, y_train)

y_pred = Bag_regressor.predict(X_test)
test_error = mean_squared_error(y_test, y_pred)
print("The MSE for Bagging Regression was determined to be " + repr(test_error))
```

```
The MSE for Bagging Regression was determined to be 0.24860748320157489
```

In [ ]:
```

```