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.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
```

In [2]:

```
df = pd.read_csv('Carseats.csv')
df_copy = df.copy()
df_copy = df_copy.drop('Unnamed: 0', 1)

# Quantify Qualitative variables
df_copy['Urban'] = df_copy['Urban'].replace({'Yes': 1.0, 'No': -1.0})
df_copy['US'] = df_copy['US'].replace({'Yes': 1.0, 'No': -1.0})
df_copy['ShelveLoc'] = df_copy['ShelveLoc'].replace({'Good': 3.0, 'Medium': 2.0, 'Bad': 1.0})

df_copy.head()
```

```
C:\Users\ADAMYA~1\AppData\Local\Temp/ipykernel_9488/3083406175.py:3: FutureWarning: In a future version of pandas all arg
uments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df_copy = df_copy.drop('Unnamed: 0', 1)
```

Out[2]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	1.0	42	17	1.0	1.0
1	11.22	111	48	16	260	83	3.0	65	10	1.0	1.0
2	10.06	113	35	10	269	80	2.0	59	12	1.0	1.0
3	7.40	117	100	4	466	97	2.0	55	14	1.0	1.0
4	4.15	141	64	3	340	128	1.0	38	13	1.0	-1.0

In [3]:

```
y = df_copy['Sales']
X = df_copy.iloc[:,1:]
```

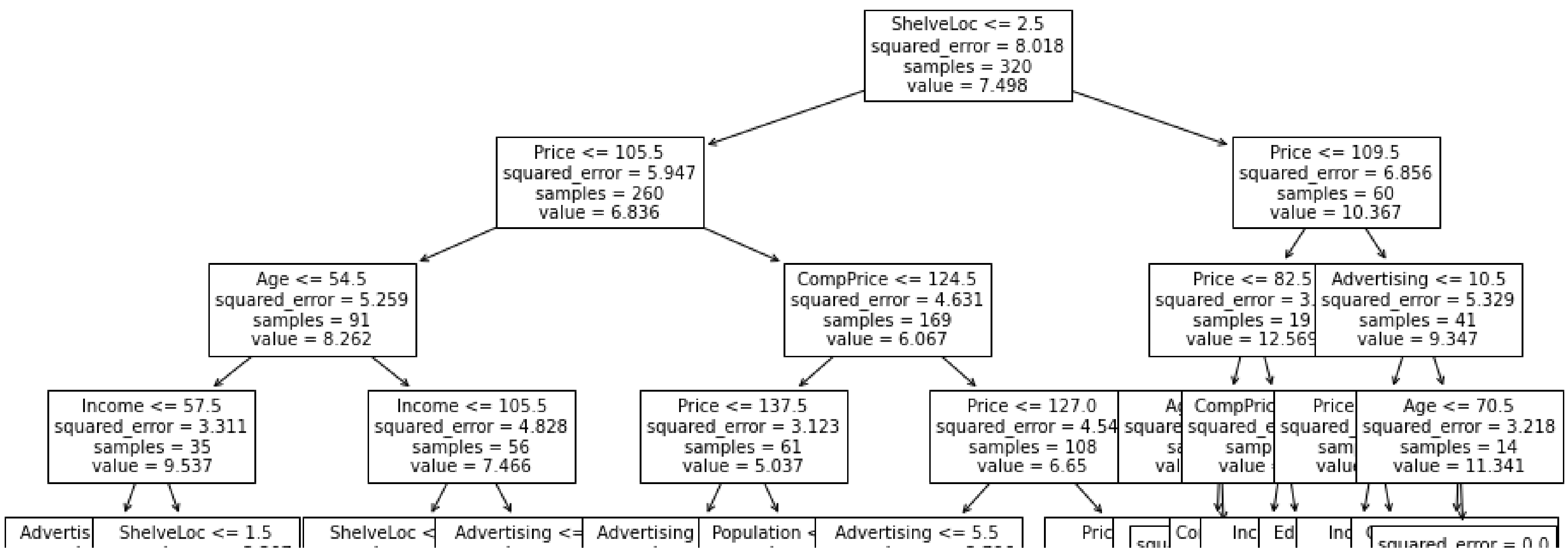
Problem a

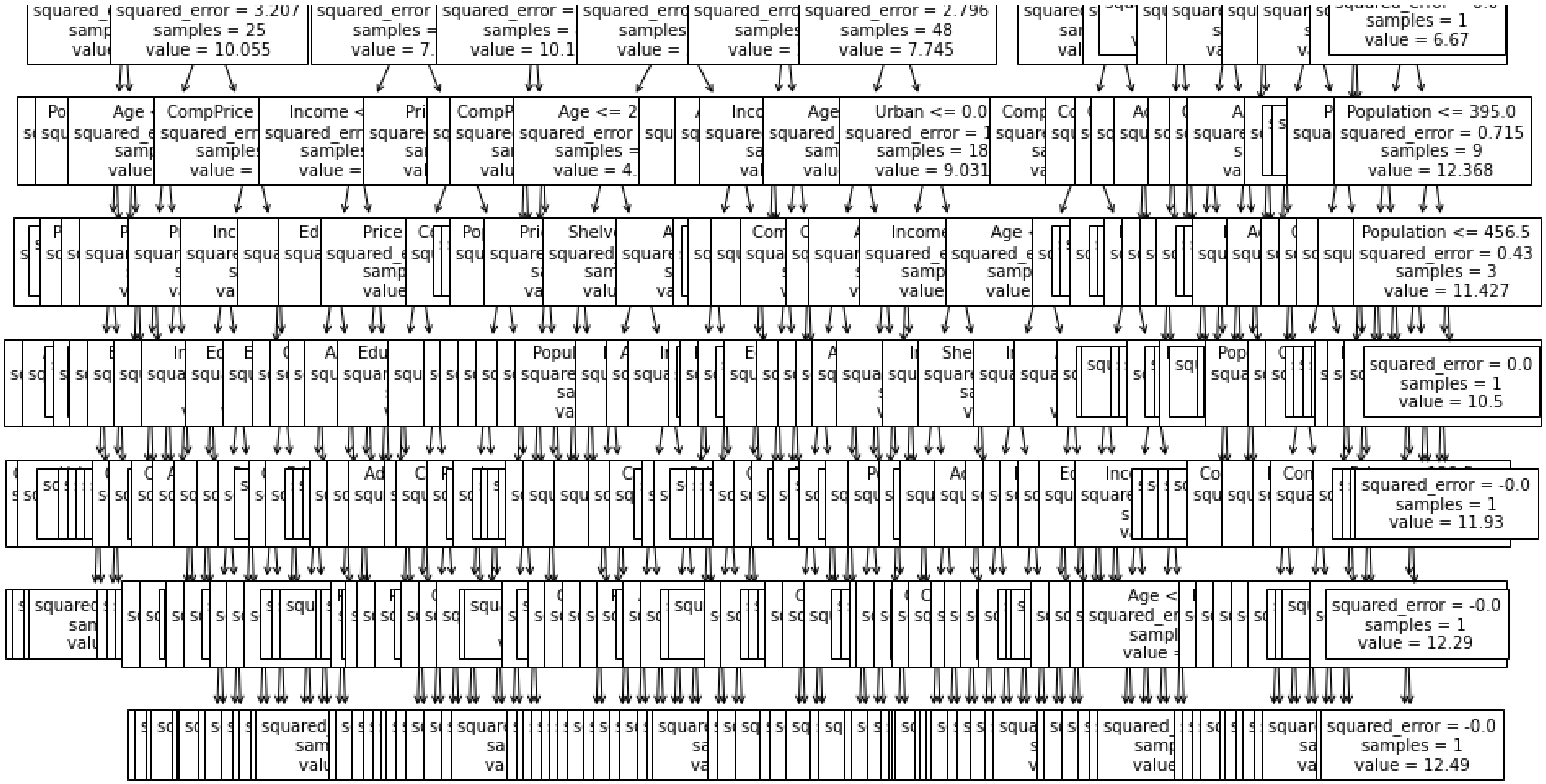
In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Problem b

In [13]:
```
from sklearn import tree
# Create Decision Tree Regressor and fit to training data
DT_regressor = DecisionTreeRegressor(max_depth=10)
DT_regressor.fit(X_train, y_train)

y_pred = DT_regressor.predict(X_test)
test_error = mean_squared_error(y_test, y_pred)
print("The MSE obtained was determined to be " + repr(test_error))

columns = ['CompPrice', 'Income', 'Advertising',
           'Population', 'Price', 'ShelveLoc',
           'Age', 'Education', 'Urban', 'US']
plt.figure(figsize=(15,15))
tree.plot_tree(DT_regressor, fontsize=10,feature_names=columns);
```

```
The MSE obtained was determined to be 3.7923133336805557
```





Problem c

In [14]:

```
from sklearn.model_selection import KFold
X_array = np.asarray(X)
best_depth = None
best_score = np.inf
depth_list = np.linspace(1,15,15)
K=10
for depth in (depth_list):
    kfold = KFold(n_splits=K, shuffle=True)

    sum_test_errors = 0
    for train_index, test_index in kfold.split(X):
    #      print("TRAIN:", train_index, "TEST:", test_index)
```

```
        X_train_curr, X_test_curr = X_array[train_index], X_array[test_index]
        y_train_curr, y_test_curr = y[train_index], y[test_index]

        DT_regressor = DecisionTreeRegressor(max_depth=depth)
        DT_regressor.fit(X_train_curr, y_train_curr)

        y_pred = DT_regressor.predict(X_test_curr)
        test_error = mean_squared_error(y_test_curr, y_pred)

        sum_test_errors = sum_test_errors + test_error

    current_test_error = sum_test_errors/K

    if current_test_error < best_score:
        best_score = current_test_error
        best_depth = depth

print("The best tree depth is " + repr(best_depth))
```

```
The best tree depth is 7.0
```

Problem d

In [23]:

```
from sklearn.ensemble import BaggingRegressor
Bag_regressor = BaggingRegressor(base_estimator=DecisionTreeRegressor(),
                                 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 obtained was determined to be " + repr(test_error))
```

```
The MSE obtained was determined to be 2.8407514
```

In [25]:

```
feature_importances = np.mean([tree.feature_importances_ for tree in Bag_regressor.estimators_], axis=0)
data = {
        'Features': list(X.columns.values),
        'Feature Importances': feature_importances
}
feat_import = pd.DataFrame(data)
feat_import
```

Out[25]:

	Features	Feature Importances
0	CompPrice	0.119675
1	Income	0.046901
2	Advertising	0.082407
3	Population	0.040547
4	Price	0.319827
5	ShelveLoc	0.248911
6	Age	0.096718
7	Education	0.030215
8	Urban	0.004147
9	US	0.010651

Problem e

In [26]:

```
# Create Decision Tree Regressor and fit to training data
RF_regressor = RandomForestRegressor()
RF_regressor.fit(X_train, y_train)

y_pred = RF_regressor.predict(X_test)
test_error = mean_squared_error(y_test, y_pred)
print("The MSE obtained was determined to be " + repr(test_error))

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

```
The MSE obtained was determined to be 2.2050548175
```

Out[26]:

Features	Feature Importances

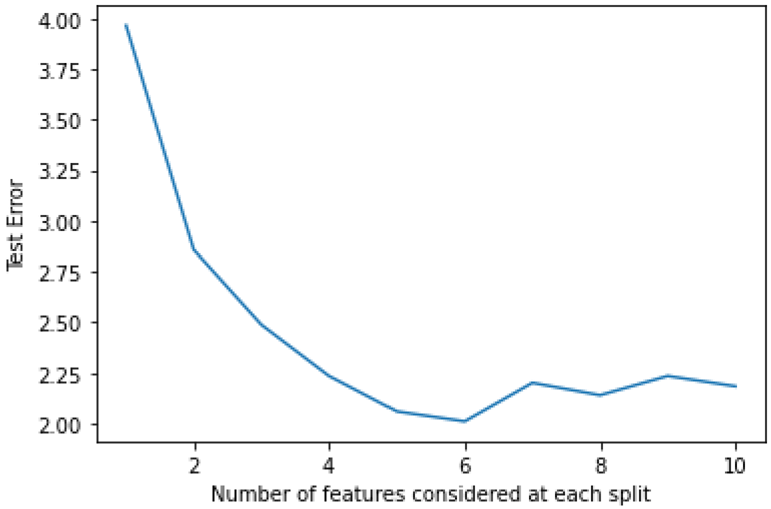
	Features	Feature Importances
0	CompPrice	0.106036
1	Income	0.049608
2	Advertising	0.094209
3	Population	0.035868
4	Price	0.300052
5	ShelveLoc	0.273900
6	Age	0.096105
7	Education	0.031598
8	Urban	0.005276
9	US	0.007348

In [27]:

```
test_error_list = []
varSplits_list = np.linspace(1,10,10).astype(int)
for split in (varSplits_list):
    RF_regressor = RandomForestRegressor(max_features=split)
    RF_regressor.fit(X_train, y_train)

    y_pred = RF_regressor.predict(X_test)
    test_error = mean_squared_error(y_test, y_pred)

    test_error_list.append(test_error)

plt.plot(varSplits_list, test_error_list)
plt.xlabel("Number of features considered at each split")
plt.ylabel("Test Error")
```

Out[27]:

```
Text(0, 0.5, 'Test Error')
```


4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
Test Error
2
4
6
8
10
Number of features considered at each split

In [ ]: