In [4]:
```
conda install mlxtend --channel conda-forge
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.7.10
  latest version: 4.8.2

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: //anaconda3

  added / updated specs:
    - mlxtend


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.8.2                |           py37_0         3.0 MB  co
nda-forge
    mlxtend-0.17.1             |             py_0         1.2 MB  co
nda-forge
    ------------------------------------------------------------
                                           Total:         4.3 MB

The following NEW packages will be INSTALLED:

  mlxtend            conda-forge/noarch::mlxtend-0.17.1-py_0

The following packages will be UPDATED:

  conda                      pkgs/main::conda-4.7.10-py37_0 --> cond
a-forge::conda-4.8.2-py37_0



Downloading and Extracting Packages
mlxtend-0.17.1       | 1.2 MB    | #################################
```

```
#### | 100%
conda-4.8.2          | 3.0 MB    | ################################
#### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [129]:

```
import numpy as np
import math
import pandas as pd
import seaborn as sb
import statsmodels.api as sm
import matplotlib.pyplot as plt
import itertools
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV, E
lasticNetCV
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
```

Conceptual Exercises

1.1

In [35]:

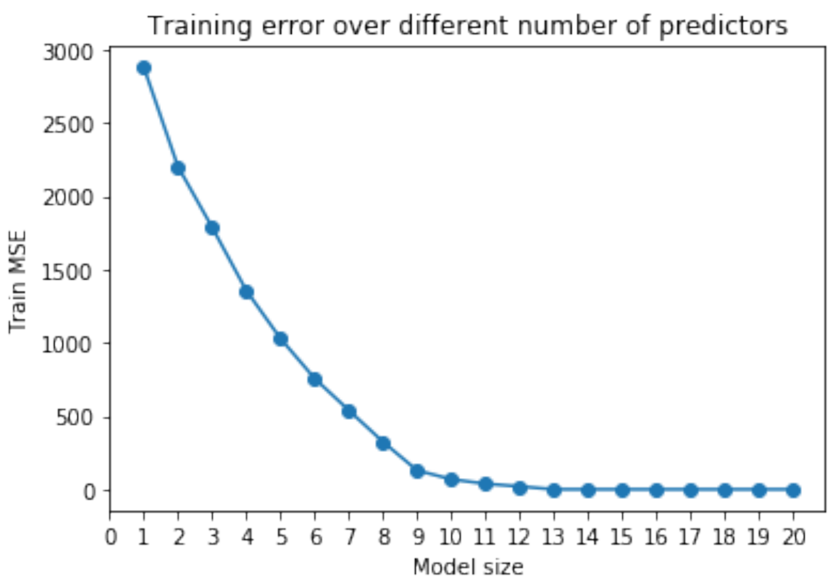
```
#set seed
np.random.seed(1234)
#simulate beta
beta = np.array([np.random.randint(-5,5) for i in range(20)])
zero = np.random.randint(1, 20, 5)
for i in zero:
    beta[i] = 0
#simulate error
err = np.random.normal(0,1, 1000)
#simulate X and generate Y
x = np.random.normal(0,5,(1000, 20))
y = np.dot(x, beta) + err
```

1.2

In [40]:
```
#split data set
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.9)
```

1.3

In [47]:
```
#perform best subset selection on the training set
k_score = []
for i in range (1,21):
    lr= LinearRegression()
    subset = sfs(lr, k_features=i, forward=True,
                 scoring ='neg_mean_squared_error', cv=0)
    subset.fit(xtrain, ytrain)
    k_score.append(-subset.k_score_)
#plot the training MSE associated with the best model of each size
plt.plot(np.arange(1,21), k_score, marker='o')
plt.xticks(np.arange(0, 21, 1))
plt.xlabel("Model size")
plt.ylabel("Train MSE")
plt.title("Training error over different number of predictors")
plt.show()
```

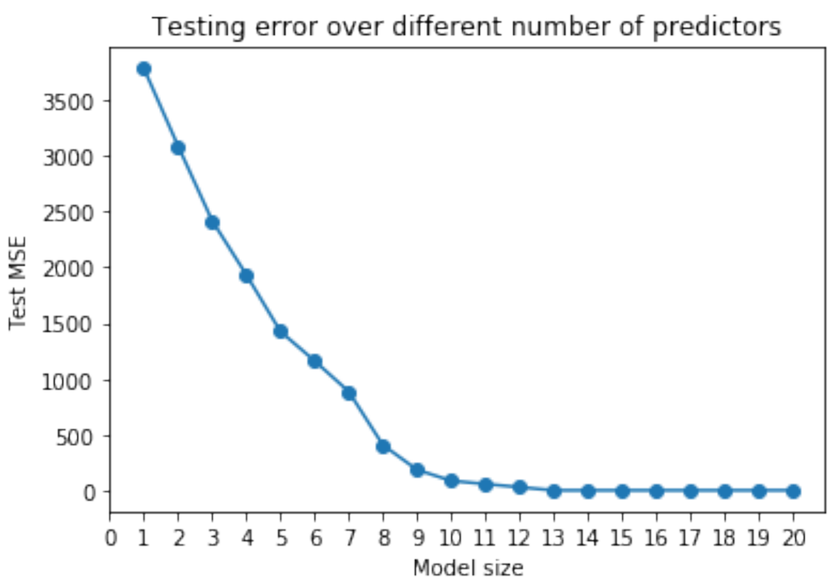


The training set MSE takes on the minimum train mse value at size 20(20 features)

1.4

In [70]:
```
#Plot the test set MSE associated with the best model of each size.
mse_test = []
feature_idx = []
models = []
for idx in range (1,21):
    lm= LinearRegression()
    subset = sfs(lm, k_features=idx, forward=True,
                    scoring ='neg_mean_squared_error', cv=0)
    subset.fit(xtrain, ytrain)
    lm = lm.fit(xtrain[:, subset.k_feature_idx_], ytrain)
    mse_test.append(mean_squared_error(lm.predict(xtest[:, subset.k_fe
ature_idx_]),ytest))
    feature_idx.append(list(subset.k_feature_idx_))
    models.append(lm)


plt.plot(np.arange(1,21), mse_test, marker='o')
plt.xticks(np.arange(0, 21, 1))
plt.xlabel("Model size")
plt.ylabel("Test MSE")
plt.title("Testing error over different number of predictors")
plt.show()
```



1.5 The testing set MSE takes on the minimum train mse value at size 13(13 features). As there are several zero beta in the model. This result shows that the best model selected valuable predictors and excolueded the useless one.

1.6

In [79]:
```
min_mse = models[13]# we knew the best model is at size 13
model_coef = list(min_mse.coef_)
print(model_coef)
```

```
[-1.988200063160434, -0.9857182802475211, 3.9755905065971464, -3.976889594347562, 1.0262253659707778, 3.018271558452329, -5.017127490090471, -0.048562900083570545, -5.033394396113445, 4.05736414619728, 0.9607432560743083, -3.0005374748017215, -5.009724901510801, -2.956013666001007]
```

In [80]:
```
true_coef = list(filter(lambda num: num != 0, beta))
print(true_coef)
```

```
[-2, -1, 4, -4, 1, 3, -5, -5, 4, 1, -3, -5, -3]
```

We knew from the previous question that the best model is at size 13. Comparing these two lists of coefficients, we can find that the best model's estimation of coefficients is very close to the true model.

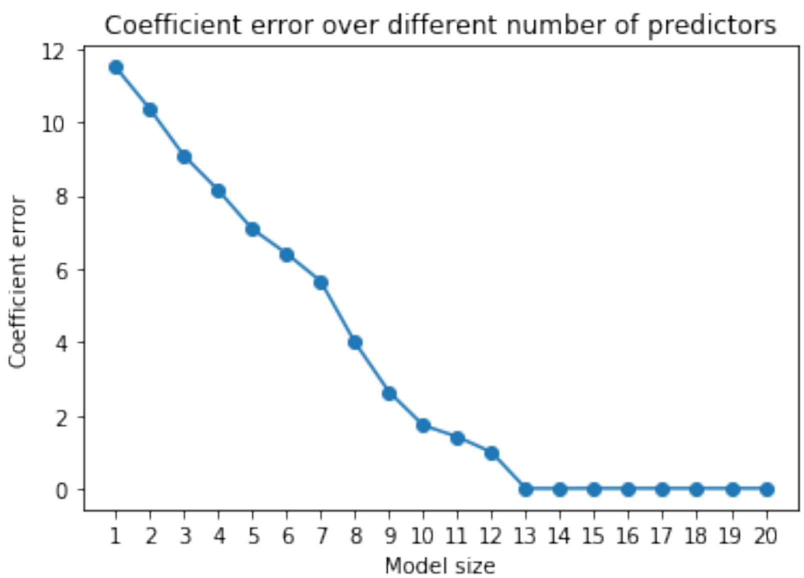
1.7

In [84]:
```
df = pd.DataFrame({"model_size": np.arange(1,21),"MSE": mse_test,"feature_index": feature_idx})
df
```

Out[84]:

	model_size	MSE	feature_index
0	1	3781.012229	[13]
1	2	3087.150657	[11, 13]
2	3	2416.948868	[11, 13, 17]
3	4	1933.259908	[5, 11, 13, 17]
4	5	1424.593965	[5, 6, 11, 13, 17]
5	6	1161.271789	[5, 6, 11, 13, 17, 19]
6	7	886.149136	[5, 6, 11, 13, 16, 17, 19]
7	8	407.863328	[5, 6, 11, 13, 14, 16, 17, 19]
8	9	186.054795	[5, 6, 10, 11, 13, 14, 16, 17, 19]
9	10	87.070855	[0, 5, 6, 10, 11, 13, 14, 16, 17, 19]
10	11	58.027627	[0, 5, 6, 9, 10, 11, 13, 14, 16, 17, 19]
11	12	29.892351	[0, 3, 5, 6, 9, 10, 11, 13, 14, 16, 17, 19]
12	13	1.318692	[0, 3, 5, 6, 9, 10, 11, 13, 14, 15, 16, 17, 19]
13	14	1.446544	[0, 3, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17...
14	15	1.455861	[0, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16,...
15	16	1.486163	[0, 1, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, ...
16	17	1.510078	[0, 1, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, ...
17	18	1.525699	[0, 1, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 1...
18	19	1.545949	[0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
19	20	1.557500	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...

In [93]:
```
# plot the coefficient errors
coef_err = []
for k in range(20):
    model_coef = np.zeros(20)
    f = df["feature_index"][k]
    model = lm.fit(x[:,f], y)
    for i, coef in zip(f,model.coef_):
        model_coef[i]=coef
    coef_err.append(np.sqrt(np.sum((beta - model_coef) ** 2)))
plt.plot(np.arange(1,21), coef_err, marker='o')
plt.xticks(np.arange(1,21))
plt.xlabel('Model size')
plt.ylabel('Coefficient error')
plt.title('Coefficient error over different number of predictors')
plt.show()
```



Comparing to the test MSE plot, the coeffient error also gets its minimum value at size 13.

Application exercises

2.1

In [94]:
```
gss_train = pd.read_csv("gss_train.csv")
gss_test = pd.read_csv("gss_test.csv")
```

In [95]:
```
x_train = gss_train.drop('egalit_scale', axis=1)
y_train = gss_train['egalit_scale']
x_test = gss_test.drop('egalit_scale', axis=1)
y_test = gss_test['egalit_scale']
```

In [106]:
```
lm = LinearRegression().fit(x_train,y_train)
pred =lm.predict(x_test)
mse = mean_squared_error(y_test, pred)
print("Test MSE for linaer regression is:", mse)
```

```
Test MSE for linaer regression is: 63.213629623014995
```

2.2

In [132]:
```
ridge = RidgeCV(cv=10).fit(x_train, y_train)
pred=ridge.predict(x_test)
mse_ridge = mean_squared_error(y_test, pred)
print('Test MSE of ridge regression model is', mse_ridge)
```

```
Test MSE of ridge regression model is 62.49920243957809
```

2.3

In [133]:
```
lasso = LassoCV(cv=10).fit(x_train, y_train)
print('Test MSE is for lasso is', mean_squared_error(y_test, lasso.pre
dict(x_test)))
print('Number of non-zero coefficients for lasso regression =', (lasso
.coef_ != 0).sum
())
```

```
Test MSE is for lasso is 62.7780157899344
Number of non-zero coefficients for lasso regression = 24
```

2.4

In [135]:
```
alphas=np.arange(0, 1.1, step=0.1)
elastic = ElasticNetCV(alphas=alphas,cv=10).fit(x_train, y_train)
pred=elastic.predict(x_test)
mse_elastic = mean_squared_error(y_test, pred)
print('The test MSE of elastic net regression model is', mse_elastic)
print("Number of nonzero coefficients for elastic net regreesion = ",
      np.array([elastic.coef_ != 0]).sum())
```

```
The test MSE of elastic net regression model is 62.5070860872212
Number of nonzero coefficients for elastic net regreesion =  40
```

2.5

There is no big difference among these regression models in terms of MSE and accuracy. All model give a test MSE around 62 to 63. Least Square Linear performs the worst with a test MSE of 63.21 while Ridge performs the best with a test MSE of 62.49. In general, we are not predicting individual's egalitarianism very well(accuracy is less than 0.4). To obtain a higher accuracy, we may use other models instead.