In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
```

데이터 전처리

In [3]:
```
data=pd.read_csv('C:/Users/Han/Desktop/dummy.csv')
data=data.drop('Unnamed: 0',axis=1)
```

In [5]:
```
# Y 변수 scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
DATA =pd.read_csv('C:/Users/Han/Desktop/DATA.csv')
y=DATA[['Q4B']]
y=scaler.fit_transform(y)
```

In [6]:
```
# X 변수 category화 (dummy 이므로 ordinal 을 주지 않아도 된다.)
X=data
col=X.select_dtypes(include=[np.int64]).columns
X[col] = X[col].astype('category')
```

In [7]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=0)
```

DecisionTree Regression

In [9]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [10]:
```
#DecisionTreeRegressor()
# criterion : 불순성의 기준을 뭐로 할지 'gini'(defalut) / 'entropy'
# 다음의 parameterm 들은 overfitting 을 해결해준다.
# max_depth : 트리의 최대 깊이.
#           : (defalut) full tree 가 될때까지 확장.
#           : 이를 이용해 사전 가지치기를 하고/ overfitting 을 해결할 수 있다.
# min_samples_split : 노드에서 가지 분리할 때 필요한 최소 sample 갯수에 제한을 준다.
#                   : (default) = 2
# min_samples_leaf : leaf 에서 가져야 할 최소 sample
#                  : (default) = 1
# max_features : Decision tree 를 만들때 사용할 수 있는 변수의 갯수 제한
#              : (default) = 총 변수 갯수 사용
```

In [31]:
```
model = DecisionTreeRegressor(max_depth=7,min_samples_split=100)
model.fit(X_train, y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=7, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=100, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

모델 성능평가

In [32]:
```
from sklearn import metrics
```

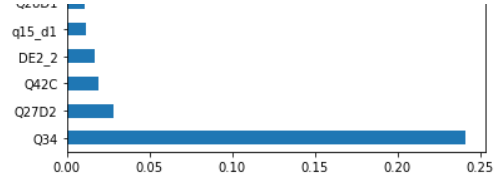
In [33]:
```
y_pred = model.predict(X_test)
print ("MSE :", metrics.mean_squared_error(y_test, y_pred))
print("R squared :", metrics.r2_score(y_test, y_pred))
```

```
MSE : 0.7774763230983701
R squared : 0.21256402111514106
```

Feature Importance

In [12]:
```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d43525f8>
```



Lasso regression

In [50]:
```
from sklearn.linear_model import LassoCV
from sklearn.linear_model import Lasso
```

Hyperparameter 설정

In [61]:
```
alphas = np.logspace(-4, 4, num=300) # 10^-4 ~ 10^4
lassocv = LassoCV(alphas = alphas, cv=5)
lassocv.fit(X_train,y_train) ;
```

```
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:1100: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:471: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 0.8665777534715744, tolerance: 0.5260560581072566
  tol, rng, random, positive)
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:471: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 1.0330270763161025, tolerance: 0.5260560581072566
  tol, rng, random, positive)
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:471: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 1.0491027198413576, tolerance: 0.5260560581072566
  tol, rng, random, positive)
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:471: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 1.0449664755856247, tolerance: 0.5260560581072566
  tol, rng, random, positive)
```

```
LassoCV(alphas=array([1.00000000e-04, 1.06354496e-04, 1.13112788e-04, 1.20300535e-04,
       1.27945027e-04, 1.36075289e-04, 1.44722187e-04, 1.53918552e-04,
       1.63699300e-04, 1.74101565e-04, 1.85164842e-04, 1.96931134e-04,
       2.09445114e-04, 2.22754295e-04, 2.36909207e-04, 2.51963593e-04,
       2.67974609e-04, 2.85003044e-04, 3.03113550e-04, 3.22374888e-04,
       3.42860186e-04, 3.64647222e-0...
       5.07791724e+03, 5.40059328e+03, 5.74377375e+03, 6.10876161e+03,
       6.49694260e+03, 6.90979055e+03, 7.34887289e+03, 7.81585671e+03,
       8.31251499e+03, 8.84073340e+03, 9.40251743e+03, 1.00000000e+04]),
        copy_X=True, cv=5, eps=0.001, fit_intercept=True, max_iter=1000,
        n_alphas=100, n_jobs=None, normalize=False, positive=False,
        precompute='auto', random_state=None, selection='cyclic', tol=0.0001,
        verbose=False)
```

In [62]:
```
# 최적의 alpha = 0.003150
lassocv.alpha_
```

```
0.0031501247957553278
```

model fitting

In [66]:
```
model = Lasso(alpha = 0.003150)
model.fit(X_train,y_train)
```

```
Lasso(alpha=0.00315, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

모델 성능평가

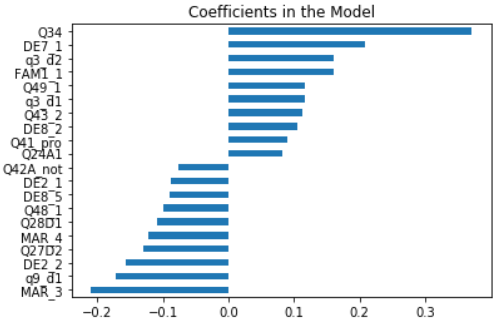
In [67]:
```
predicted = model.predict(X_test)

print ("MSE :", metrics.mean_squared_error(y_test, predicted))
print('R_squared :',model.score(X_test, y_test)) # C-V 로 찾은 최적의 ridge 로 계산한 R^2
```

```
MSE : 0.6844895462342775
R_squared : 0.3067419806078835
```

Feature Importance

In [72]:
```
coef = pd.Series(model.coef_, index = X_train.columns).sort_values()
imp_coef = pd.concat([coef.head(10), coef.tail(10)])
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Model")
```

```
Text(0.5, 1.0, 'Coefficients in the Model')
```



RandomForeset Regression

In [76]:
```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=10, max_leaf_nodes=16, random_state=42)
model.fit(X_train, y_train)
```

```
C:\Users\Han\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=16,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

모델 성능평가

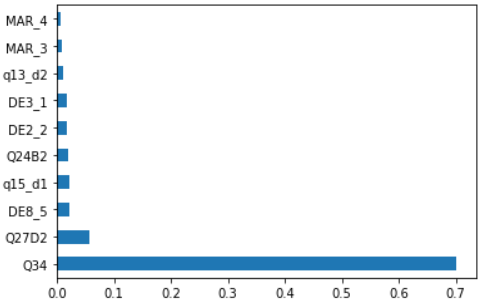
In [81]:
```
y_pred = model.predict(X_test)
print ("MSE :", metrics.mean_squared_error(y_test, y_pred))
print('R_squared :',model.score(X_test, y_test))
```

```
MSE : 0.738471306178806
R_squared : 0.25206870153688987
```

Feature Importance

In [80]:
```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1dae55198>
```



Gradient Boosting resgressoion

In [85]:
```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor(n_estimators=10, learning_rate=1.0, random_state=42)
model.fit(X_train, y_train)
```

```
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\ensemble\gradient_boosting.py:1450: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=1.0, loss='ls', max_depth=3,
```

```
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=10,
n_iter_no_change=None, presort='auto',
random_state=42, subsample=1.0, tol=0.0001,
validation_fraction=0.1, verbose=0, warm_start=False)
```

모델 성능평가

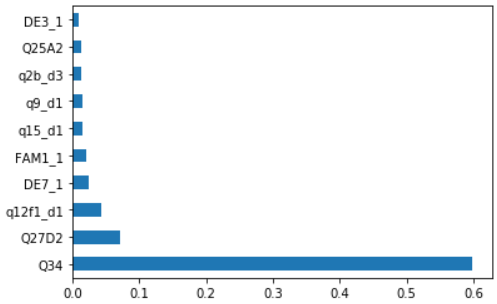
In [86]:
```
y_pred = model.predict(X_test)
print ("MSE :", metrics.mean_squared_error(y_test, y_pred))
print('R_squared :',model.score(X_test, y_test))
```

```
MSE : 0.7387107423492368
R_squared : 0.25182619813242313
```

Feature Importance

In [88]:
```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1daa48978>
```



AdaBoost Regression

In [92]:
```
from sklearn.ensemble import AdaBoostRegressor
model =AdaBoostRegressor(n_estimators=10, learning_rate=1.0, random_state=42)
model.fit(X_train, y_train)
```

```
C:\Users\Han\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please cha
nge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
AdaBoostRegressor(base_estimator=None, learning_rate=1.0, loss='linear',
                  n_estimators=10, random_state=42)
```

모델 성능평가

In [93]:
```
y_pred = model.predict(X_test)
print ("MSE :", metrics.mean_squared_error(y_test, y_pred))
print('R_squared :',model.score(X_test, y_test))
```

```
MSE : 0.8038143649977205
R_squared : 0.18588858266284825
```

Feature Importance

In [94]:
```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1dad04cc0>
```

Voting Regression

In [90]:
```
#얘는 아직 어떤 모델을 넣어야할지 몰라서 보류..,.
#from sklearn.ensemble import VotingRegressor
```

...

1. 행복도를 classify 해서 (10점 기준으로 많이 끊기는 모습 때문) regression 이 아닌 classification 으로 문제를 다시 풀 수 있을까?
2. 거의 모든 문항에서 q34 가 매우 큰 importance 를 나타내고 있는데, 이를 제거해보고 다시 분석해봐야하나?

In [ ]: