Team: Gamestopper

In [1]:
```
# Investor Risk Tolerance and Robo advisors
# The goal of this case study is to build a machine learning model to predict
```

In [2]:
```
## Content
```


1. Problem Definition

In the supervised regression framework used for this case study, the predicted variable is the “true” risk tolerance of an individual10 and the predictor variables are demo- graphic, financial and behavioral attributes of an individual

For this case study the data used is from survey of Consumer Finances which is conducted by the Federal Reserve Board. The data source is :
https://www.federalreserve.gov/econres/scf_2009p.htm

2. Getting Started- Loading the data and python packages

2.1. Loading the python packages

In [3]:

```
import numpy as np
import pandas as pd
import pandas_datareader as web
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import seaborn as sns##### Core packages

##### Machine Learning packages
#Keras
#tensorflow
#tensorboard
# If you have a TF-compatible GPU and you want to enable GPU support, then
# replace tensorflow with tensorflow-gpu.

##### Statistics packages
#statsmodels

##### Data Download packages
#Quandl
#xlrd
#yfinance

##### Additional Visualization packages
#plotly


#The following additional packages are also used in the case studies.
#However, the following packages are commented as the installation is time ta
#Uncomment these in case you want to install them from the requirement.txt fi

# Additional utilities used for robo-advisor team technical case study
#dash
#dash-core-components
#dash-daq
#cvxopt

import copy
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

```
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.neural_network import MLPRegressor

#Libraries for Deep Learning Models
# from keras.models import Sequential
# from keras.layers import Dense
# from keras.optimizers import SGD
# from keras.layers import LSTM
# from keras.wrappers.scikit_learn import KerasRegressor

#Libraries for Statistical Models
import statsmodels.api as sm

#Libraries for Saving the Model
from pickle import dump
from pickle import load
```

2.2. Loading the Data

In [4]:
```
# load dataset
dataset = pd.read_excel('SCFP2009panel.xlsx')
```

In [5]:
```
#Diable the warnings
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
type(dataset)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
dataset.shape
```

Out[7]:
```
(19285, 515)
```

In [8]:
```
# <a id='2'></a>
## 3. Data Preparation and Feature Selection
```

3.1. Preparing the predicted variable

The dataset from "Survey of Consumer Finances" contains the Household's demographics, net worth, financial and non-financial assets for the same demographics in 2007 (pre-crisis) and 2009(post-crisis).

We prepare the predicted variable, which is the "true" risk tolerance in the following steps. There are different ways of getting the "true" risk tolerance. The idea and the purpose of this case study is to come up with an approach to solve the behavioral finance problem using machine learning.

The steps to compute the predicted variables are as follows:

1) Compute the Risky asset and the riskless assets for all the individuals in the survey data. Risky and riskless assets are defined as follows:

- **Risky assets** is investments in mutual funds, stocks, bonds, commodities, and real estate, and an estimate of human capital.
- **Risk Free Assets**: checking and savings balances,certificates of deposit, and other cash balances and equivalents.

2) We take the ratio of risky assets to total assets of an investor and consider that as a measure of risk tolerance of an investor. From the data of SCF, we have the data of risky and riskless assets for the individuals for 2007 and 2009. We use this data and normalise the risky assets with the stock price of 2007 vs. 2009 to get risk tolerance.

- **Risk Tolerance** just defined as the ratio of Risky Asset to Riskless Assets normalised with the average S&P500 of 2007 vs 2009. Average S&P500 in 2007: 1478 Average S&P500 in 2009: 948

3) In a lot of literature, an intelligent investor is the one who doesn't change its risk tolerance during the change in the market. So, we consider the investors who change their risk tolerance by less than 10% between 2007 and 2009 as the intelligent investors. Ofcourse this is a qualitative judgement and is subject to change. However, as mentioned before more than being accurate and precise the purpose of theis case study is to demonstrate the usage of the machine learning and provide a machine learning based framework in behavioral finance and portfolio management which can be further leveraged for more detailed analysis.

In [9]:
```
#Average SP500 during 2007 and 2009
Average_SP500_2007=1478
Average_SP500_2009=948

#Risk Tolerance 2007
dataset['RiskFree07']= dataset['LIQ07'] + dataset['CDS07'] + dataset['SAVBND0
dataset['Risky07'] = dataset['NMMF07'] + dataset['STOCKS07'] + dataset['BOND0
dataset['RT07'] = dataset['Risky07']/(dataset['Risky07']+dataset['RiskFree07'

#Risk Tolerance 2009
dataset['RiskFree09']= dataset['LIQ09'] + dataset['CDS09'] + dataset['SAVBND0
dataset['Risky09'] = dataset['NMMF09'] + dataset['STOCKS09'] + dataset['BOND0
dataset['RT09'] = dataset['Risky09']/(dataset['Risky09']+dataset['RiskFree09'
                (Average_SP500_2009/Average_SP500_2007)
```

In [10]:
```
dataset2 = copy.deepcopy(dataset)
dataset.head()
```

Out[10]:

	YY1	Y1	WGT09	AGE07	AGECL07	EDUC07	EDCL07	MARRIED07	KIDS07	LIFECL0
0	1	11	11668.134198	47	3	12	2	1	0	
1	1	12	11823.456494	47	3	12	2	1	0	
2	1	13	11913.228354	47	3	12	2	1	0	
3	1	14	11929.394266	47	3	12	2	1	0	
4	1	15	11917.722907	47	3	12	2	1	0	

5 rows × 521 columns

Let us compute the percentage change in risk tolerance between 2007 and 2009.

In [11]:
```
dataset2['PercentageChange'] = np.abs(dataset2['RT09']/dataset2['RT07']-1)
```

In [12]:
```
#Checking for the rows with null or nan values and removing them.
```

In [13]:
```
#Checking for any null values and removing the null values'''
print('Null Values =',dataset2.isnull().values.any())
```

```
Null Values = True
```

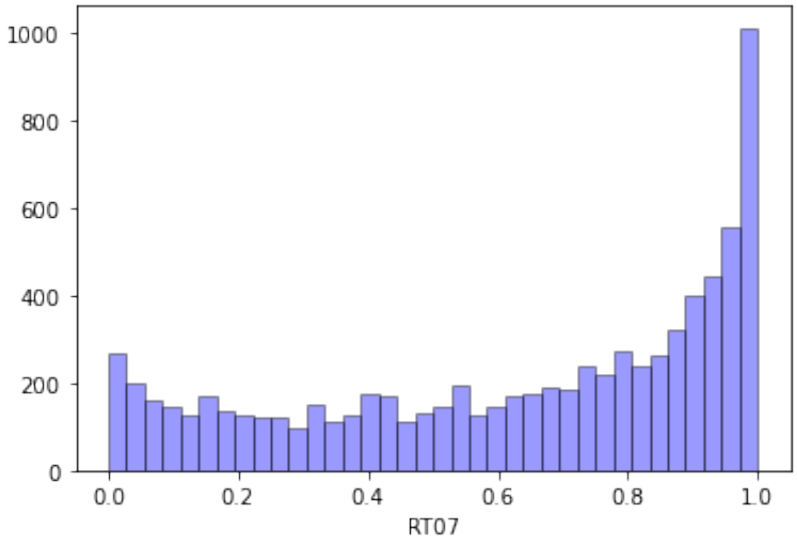
In [14]:
```
# Drop the rows containing NA
dataset2=dataset2.dropna(axis=0)

dataset2=dataset2[~dataset2.isin([np.nan, np.inf, -np.inf]).any(1)]

#Checking for any null values and removing the null values'''
print('Null Values =',dataset2.isnull().values.any())
```

```
Null Values = False
```

Let us plot the risk tolerance of 2007 and 2009.

In [15]:
```
sns.distplot(dataset2['RT07'], hist=True, kde=False,
             bins=int(180/5), color = 'blue',
             hist_kws={'edgecolor':'black'})
```

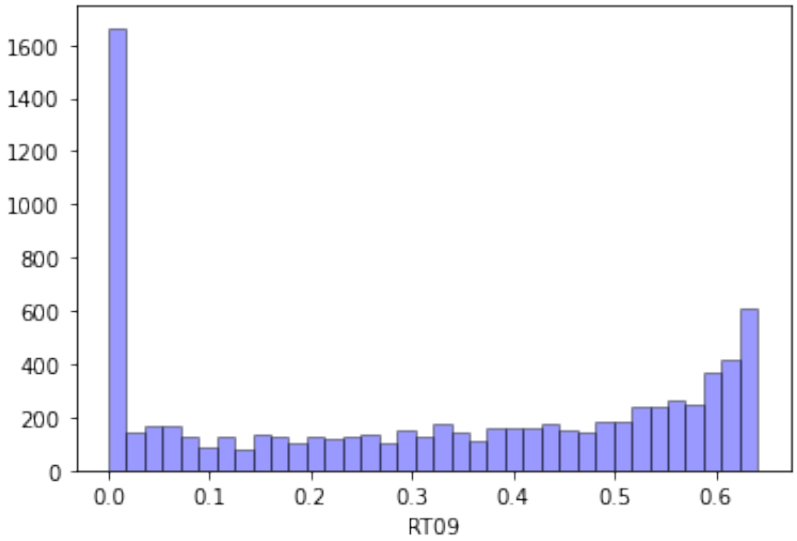
Out[15]:
```
<AxesSubplot:xlabel='RT07'>
```



Looking at the risk tolerance of 2007, we see that a significant number of individuals had risk tolerance close to one.Meaning the investment ws more skewed towards the risky assets as compared to the riskless assets. Now let us look at the risk tolerance of 2009.

In [16]:
```
sns.distplot(dataset2['RT09'], hist=True, kde=False,
             bins=int(180/5), color = 'blue',
             hist_kws={'edgecolor':'black'})
```

Out[16]: <AxesSubplot:xlabel='RT09'>



dataset3 = copy.deepcopy(dataset2)

Clearly, the behavior of the individuals reversed in 2009 after crisis and majority of the investment was in risk free assets. Overall risk tolerance decreased, which is shown by majority of risk tolerance being close to 0 in 2009. In the next step we pick the intelligent investors whose risk tolerance change between 2007 and 2009 was less than 10%

In [17]:
```
dataset3 = copy.deepcopy(dataset2)
dataset3 = dataset3[dataset3['PercentageChange']<=.1]
```

We assign the true risk tolerance as the average risk tolerance of these intelligent investors between 2007 and 2009. This is the predicted variable for this case study. The purpose would be to predict the true risk tolerance of an individuals given the demographic, financial and willingness to take risk related features.

In [18]:
```
dataset3['TrueRiskTolerance'] = (dataset3['RT07'] + dataset3['RT09'])/2
```

Let us drop other labels which might not be needed for the prediction.

In [19]:
```
dataset3.drop(labels=['RT07', 'RT09'], axis=1, inplace=True)
dataset3.drop(labels=['PercentageChange'], axis=1, inplace=True)
```

3.2. Feature Selection-Limit the Feature Space

3.2.2. Features elimination

In order to filter the features further we do the following:

1. Check the description in the Data Dictionary (https://www.federalreserve.gov/econres/files/codebk2009p.txt, https://www.federalreserve.gov/econresdata/scf/files/fedstables.macro.txt)and only keep the features that are intuitive The description is as follows:

- AGE: There are 6 age categories, where 1 represents age less than 35 and 6 represents age more than 75.
- EDUC: There are 4 education categories, where 1 represents no high school and 4 represents college degree.
- MARRIED: It represents marital status. There are two categories where 1 represents married and 2 represents unmarried.
- OCCU: It represents occupation category. 1 represents managerial category and 4 represents unemployed.
- KIDS: It represents number of kids.
- NWCAT: It represents net worth category. There are 5 categories, where 1 net worth less than 25 percentile and 5 represents net worth more than 90th percentile.
- INCCL: It represents income category. There are 5 categories, where 1 income less than 10,000 and 5 represents net worth more than 100,000
- RISK: It represents the willingness to take risk on a scale of 1 to 4, where 1 represents highest level of willingness to take risk.

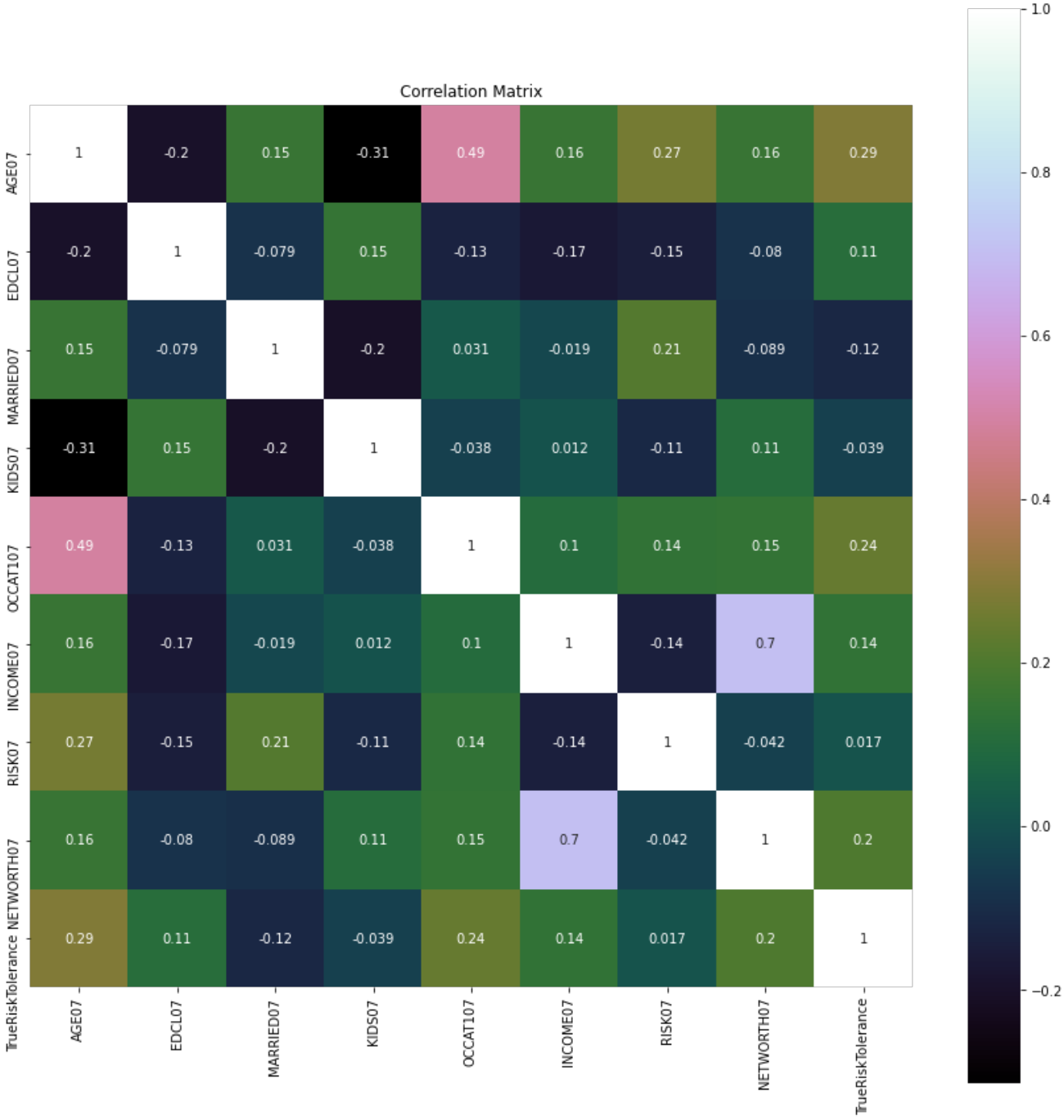
1. Keep only the intuitive factors as of 2007 only and remove all the intermediate features and features related to 2009, as the variables of 2007 are the only ones required for predicting the risk tolerance.

In [20]:
```
keep_list2 = ['AGE07','EDCL07','MARRIED07','KIDS07','OCCAT107','INCOME07','RI
]

drop_list2 = [col for col in dataset3.columns if col not in keep_list2]

dataset3.drop(labels=drop_list2, axis=1, inplace=True)
```

Let us look at the correlation among the features.

In [21]:
```
# correlation
correlation = dataset3.corr()
plt.figure(figsize=(15,15))
plt.title('Correlation Matrix')
sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='cubehelix')
```

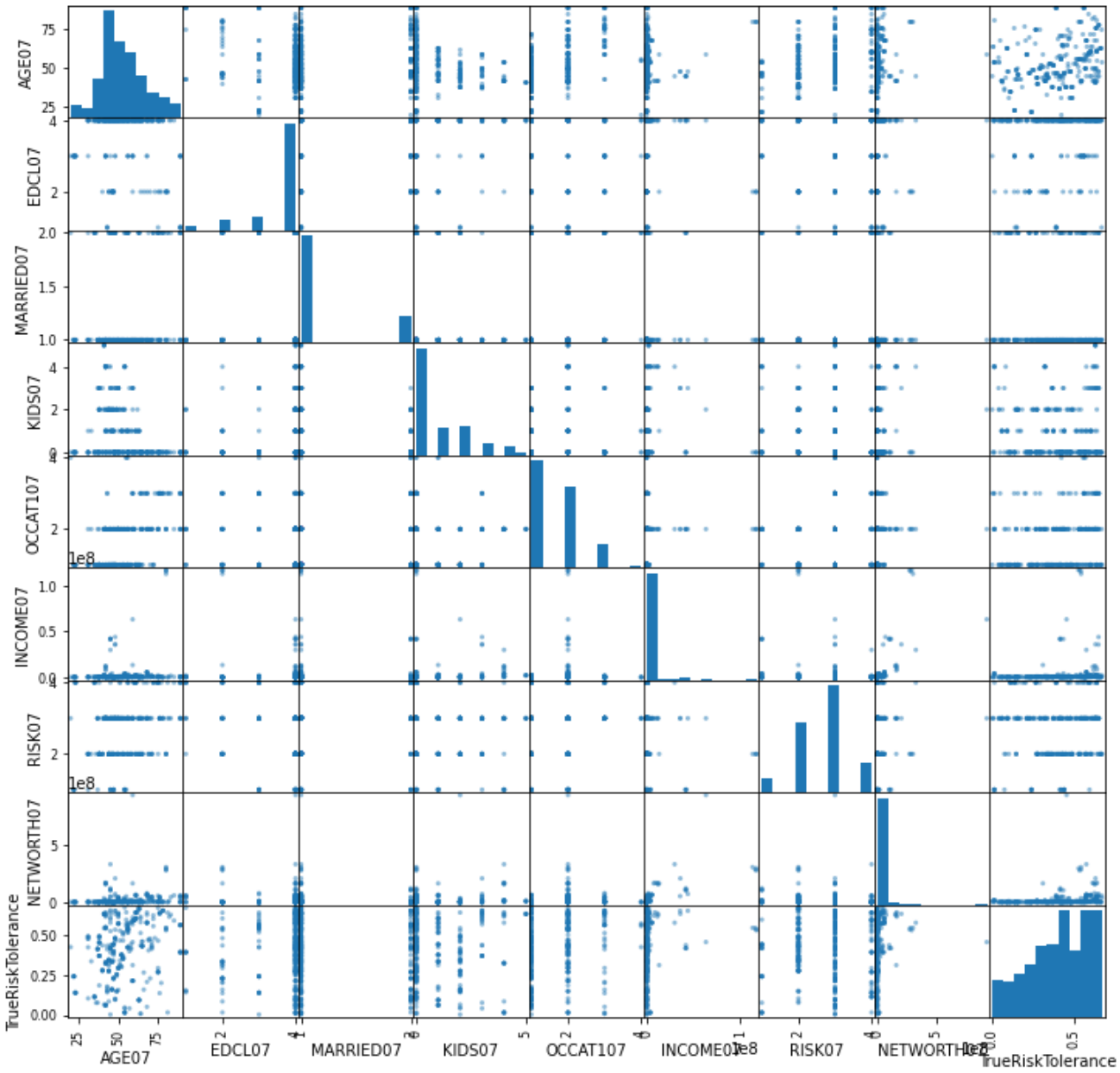
Out[21]: <AxesSubplot:title={'center':'Correlation Matrix'}>



In [22]:

```
# Scatterplot Matrix
from pandas.plotting import scatter_matrix
plt.figure(figsize=(15,15))
scatter_matrix(dataset3,figsize=(12,12))
plt.show()
```

<Figure size 1080x1080 with 0 Axes>



Looking at the correlation chart above, networth and income are positively correlated with the risk tolerance. With more number of kids and marriage the risk tolerance decreases. As the willingness to take risk decreases the risk tolerance decreases. With age there is a positive relationship of the risk tolerance.

As per the paper "Does Risk Tolerance Decrease With Age?(Hui Wang1,Sherman Hanna)", Relative risk aversion decreases as people age (i.e., the proportion of net wealth invested in risky assets increases as people age) when other variables are held constant.Therefore, risk tolerance increases with age.

So, in summary all the variables and their relationship with risk tolerance seems intuitive.

4. Evaluate Algorithms and Models

Let us evaluate the algorithms and the models.

4.1. Train Test Split

Performing a train and test split in this step.

In [23]:
```
# split out validation dataset for the end
Y= dataset3["TrueRiskTolerance"]
X = dataset3.loc[:, dataset3.columns != 'TrueRiskTolerance']
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
validation_size = 0.2
seed = 7
#X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_s

#In case the data is not dependent on the time series, then train and test sp
#This can be done by selecting an arbitrary split point in the ordered list o

train_size = int(len(X) * (1-validation_size))
X_train, X_validation = X[0:train_size], X[train_size:len(X)]
Y_train, Y_validation = Y[0:train_size], Y[train_size:len(X)]
```

4.2. Test Options and Evaluation Metrics

In [24]:
```
# test options for regression
num_folds = 10
#scoring = 'neg_mean_squared_error'
#scoring ='neg_mean_absolute_error'
scoring = 'r2'
```

4.3. Compare Models and Algorithms

Regression Models

In [25]:
```
# spot check the algorithms
models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor(random_state=seed)))
models.append(('SVR', SVR()))
#Neural Network
models.append(('MLP', MLPRegressor(random_state=seed)))
#Ensable Models
# Boosting methods
models.append(('ABR', AdaBoostRegressor(random_state=seed)))
models.append(('GBR', GradientBoostingRegressor(random_state=seed)))
# Bagging methods
models.append(('RFR', RandomForestRegressor(random_state=seed)))
models.append(('ETR', ExtraTreesRegressor(random_state=seed)))
```

K-folds cross validation

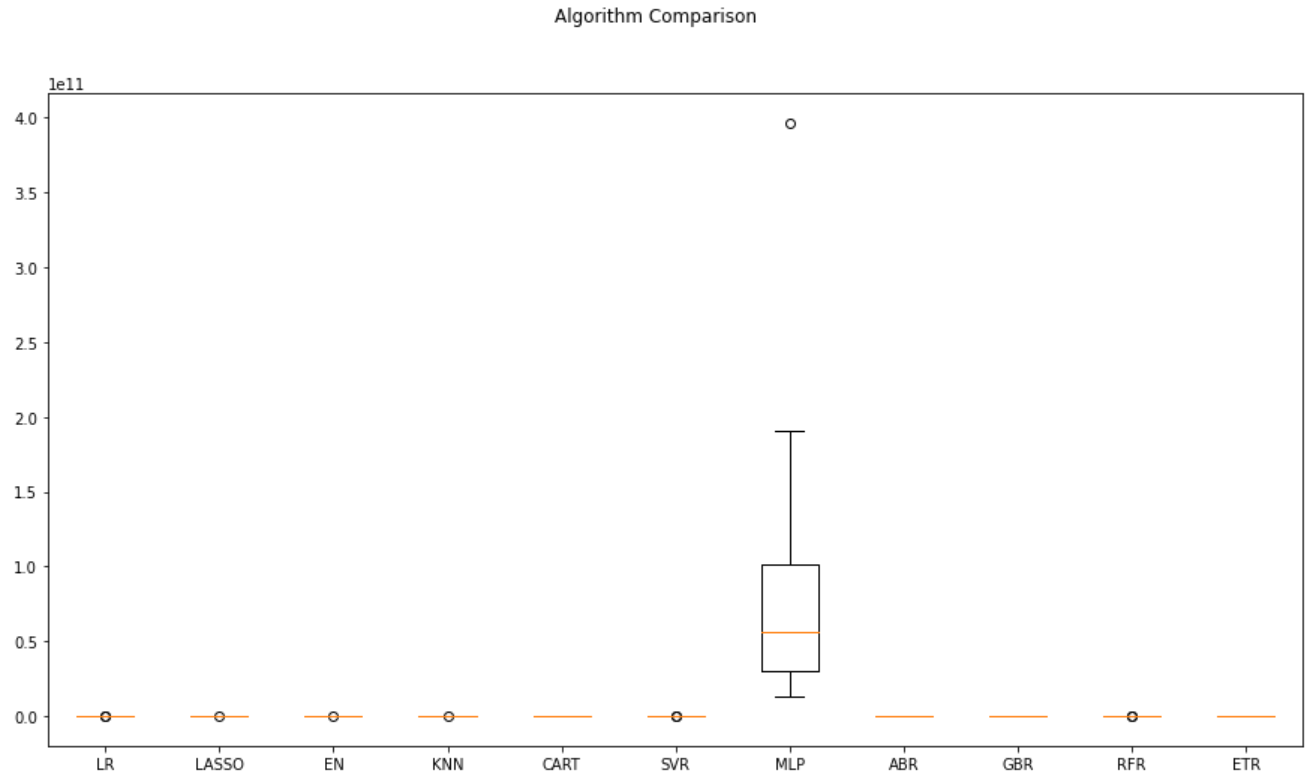
In [26]:
```
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=num_folds, shuffle=True, random_state=seed)
    #converted mean square error to positive. The lower the beter
    cv_results = -1* cross_val_score(model, X_train, Y_train, cv=kfold, scori
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.174921 (0.634490)
LASSO: 0.227683 (0.630009)
EN: 0.227312 (0.629506)
KNN: -0.538051 (0.160519)
CART: -0.748707 (0.113962)
SVR: -0.074475 (0.163281)
MLP: 98827749571.182465 (111085701692.928848)
ABR: -0.445920 (0.125358)
GBR: -0.715963 (0.073370)
RFR: -0.800866 (0.053695)
ETR: -0.831790 (0.058266)
```

Algorithm comparison

In [27]:

```
# compare algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
fig.set_size_inches(15,8)
plt.show()
```



The non linear models perform better than the linear models, which means that a non linear relationship between the risk tolerance and the difference variables use to predict it. Given random forest regression is one of the best methods, we use it for further grid search.

5. Model Tuning and Grid Search

Given that the Random Forest is the best model, Grid Search is performed on Random Forest.

In [28]:
```
# 8. Grid search : RandomForestRegressor
'''
n_estimators : integer, optional (default=10)
    The number of trees in the forest.
'''
param_grid = {'n_estimators': [50,100,150,200,250,300,350,400]}
model = RandomForestRegressor(random_state=15)
kfold = KFold(n_splits=num_folds, shuffle=True, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring,
grid_result = grid.fit(X_train, Y_train)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.795494 using {'n_estimators': 250}
0.791193 (0.037596) with: {'n_estimators': 50}
0.788716 (0.042772) with: {'n_estimators': 100}
0.789934 (0.044191) with: {'n_estimators': 150}
0.794519 (0.044775) with: {'n_estimators': 200}
0.795494 (0.046474) with: {'n_estimators': 250}
0.795221 (0.046249) with: {'n_estimators': 300}
0.795289 (0.047003) with: {'n_estimators': 350}
0.794760 (0.046860) with: {'n_estimators': 400}
```

Random forest with number of estimators 250, is the best model after grid search.

6. Finalise the Model

Finalize Model with best parameters found during tuning step.

6.1. Results on the Test Dataset

In [29]:
```
# prepare model
model = RandomForestRegressor(n_estimators = 250)
model.fit(X_train, Y_train)
```

Out[29]:
```
RandomForestRegressor(n_estimators=250)
```

In [30]:
```
from sklearn.metrics import r2_score
predictions_train = model.predict(X_train)
print(r2_score(Y_train, predictions_train))
```

```
0.9698739482007273
```

In [31]:
```
# estimate accuracy on validation set
# transform the validation dataset
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
#rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(X_validation)
print(mean_squared_error(Y_validation, predictions))
print(r2_score(Y_validation, predictions))
```

```
0.027988167736243942
0.09529312892363406
```

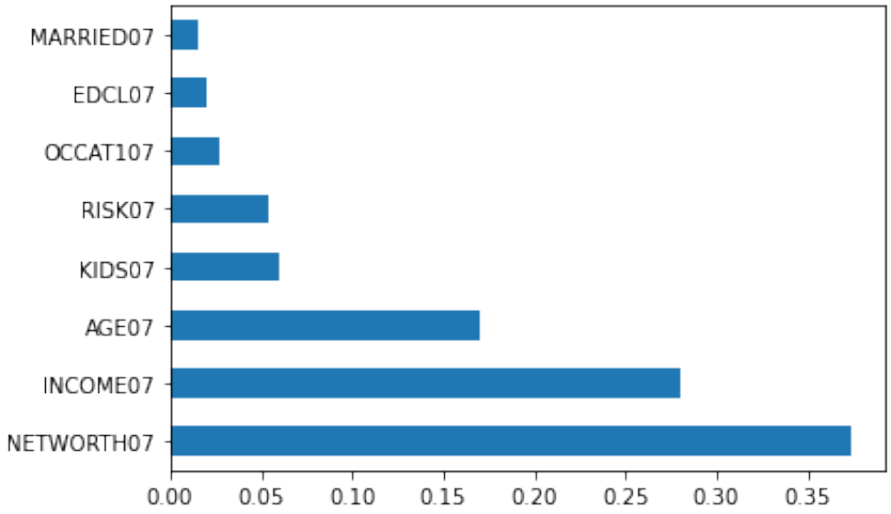
From the mean square error and R2 shown above for the test set, the results look good.

6.2. Feature Importance and Features Intuition

Looking at the details above Random forest be worthy of further study. Let us look into the Feature Importance of the RF model

In [32]:
```
import pandas as pd
import numpy as np
model = RandomForestRegressor(n_estimators= 200,n_jobs=-1,random_state=seed)
model.fit(X_train,Y_train)
print(model.feature_importances_) #use inbuilt class feature_importances of t
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
[0.17045043 0.01983408 0.0151206  0.05964268 0.0268864  0.28021067
 0.05410668 0.37374846]
```



From the chart above, income and networth followed by age and willingness to take risk are the key variables to decide the risk tolerance. These variables have been considered as the key variables to model the risk tolerance across several literature.

6.3. Save Model for Later Use

In [33]:
```
# Save Model Using Pickle
from pickle import dump
from pickle import load

# save the model to disk
filename = 'finalized_model.sav'
dump(model, open(filename, 'wb'))
```

In [34]:
```
# load the model from disk
loaded_model = load(open(filename, 'rb'))
# estimate accuracy on validation set
predictions = loaded_model.predict(X_validation)
result = mean_squared_error(Y_validation, predictions)
print(r2_score(Y_validation, predictions))
print(result)
```

```
0.09917027674682599
0.027868223622760417
```

Conclusion:

We showed that machine learning models might be able to objectively analyze the behavior of different investors in a changing market and attribute these changes to variables involved in determining risk appetite. With an increase in the volume of investor's data and availability of rich machine learning infrastructure, such models might prove to be more useful.

We saw that there is a non-linear relationship between the variables and the risk tolerance. Income and net worth followed by age and willingness to take risk are the key variables to decide the risk tolerance. These variables have been considered as the key variables to model the risk tolerance across several literature.