```
import pandas as pd
cols = ['user_id', 'movie_id', 'rating', 'timestamp']
ratings =  pd.read_csv('../stat_pandas_ratings/ratings.dat', sep='::',header=None,name
cols = ['movie_id', 'title', 'genres']
movies =  pd.read_csv('../stat_pandas_ratings/movies.dat',sep='::',header=None,names=
cols = ['user_id', 'gender', 'age', 'occupation', 'zip']
users = pd.read_csv('../stat_pandas_ratings/users.dat', sep='::', header=None,names=co
```

```
genre_iter = (set(x.split('|')) for x in movies.genres)
genres = sorted(set.union(*genre_iter))
dummies = pd.DataFrame(np.zeros((len(movies), len(genres))), columns=genres)
for i, gen in enumerate(movies.genres):
   dummies.ix[i, gen.split('|')] = 1
movies_windic = movies.join(dummies.add_prefix('Genre_'))
movies_windic = movies_windic.drop(['title','genres'],axis=1)
```

```
joined = ratings.merge(movies_windic, left_on='movie_id',right_on='movie_id')
```

```
y = joined.groupby('user_id').sum()
y = y.drop(['movie_id','rating','timestamp'],axis=1)
y[y > 0.0] = 1.0
```

```
from sklearn.feature_extraction import DictVectorizer
def one_hot_dataframe(data, cols):
    vec = DictVectorizer()
    mkdict = lambda row: dict((col, row[col]) for col in cols)
    tmp = vec.fit_transform(data[cols].to_dict(outtype='records')).toarray()
    vecData = pd.DataFrame(tmp)
    vecData.columns = vec.get_feature_names()
    vecData.index = data.index
    data = data.drop(cols, axis=1)
    data = data.join(vecData)
    return data

X = users.copy()
X['occupation2'] = users['occupation'].map(lambda x: str(x))
X['zip2'] = users['zip'].map(lambda x: str(x)[0])
X['zip3'] = users['zip'].map(lambda x: str(x)[:2])
X = one_hot_dataframe(X,['occupation2','gender','zip2','zip3'])
X = X.drop(['occupation','zip'],axis=1)
X = X.set_index('user_id')
X = X.ix[y.index]
X = X.reindex(y.index)
print X.shape, y.shape
```

```
(6040, 134) (6040, 18)
```

```
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Lasso, Ridge, LinearRegression
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=1000)
clf = RandomForestRegressor(max_depth=3,n_estimators=5)

clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

fpr, tpr, thresholds = roc_curve(np.ravel(y_test), np.ravel(y_pred))
roc_auc = auc(fpr, tpr)
print 'Tree AUC', roc_auc

imps = pd.Series(list(clf.feature_importances_),index=X.columns)
imps = imps.order(ascending=False).head(20)
print 'important features'
print np.array(imps.index)
```

```
Tree AUC 0.844777794213
important features
['gender=F' 'age' 'gender=M' 'occupation2=10' 'zip3=46' 'zip2=3' 'zip3=37'
 'zip3=65' 'zip3=31' 'zip3=33' 'zip3=41' 'zip3=39' 'zip3=67' 'zip3=01'
 'zip3=20' 'zip3=24' 'zip3=95' 'zip3=08' 'zip3=09' 'zip3=03']
```

```
y_naive = np.array(y_train).mean(axis=0)
y_naive = pd.DataFrame([y_naive], index=range(y_test.shape[0]),
                       columns=list(y.columns))
fpr, tpr, thresholds = roc_curve(np.ravel(y_test), np.ravel(y_naive))
roc_auc = auc(fpr, tpr)
print 'Naive AUC', roc_auc
```

```
Naive AUC 0.833752370725
```

http://upload.wikimedia.org/wikipedia/commons/2/24/ZIP_Code_zones.svg