Problem 0 & 1.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [39]:
```
# read in data
train = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience - HW5
trainX = train.loc[:,'F1':]
trainY = train['Y']
testX = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience - HW5
```

In [40]:
```
realData = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience -
```

1) Model interpretability

What is the effect of MonthlyIncome to the prediction? Quantify as much as you can how 1000, 2000 or 3000 extra per month affect the probablity of deliquency. Do this by fitting a simple model on the dataset and using your best model.

In [41]:
```
from sklearn.linear_model import ElasticNetCV

import xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
```

In [42]:
```
realX = realData[realData.columns[1:]]
realY = realData[realData.columns[0]]
```

In [43]:
```
from sklearn.model_selection import cross_val_score

xgb_BOOSTER = XGBClassifier(learning_rate=0.1, n_estimators= 200, max_depth
scores = cross_val_score(xgb_BOOSTER, realX, realY, cv=5, scoring = 'roc_au
print("Avg Score: {}% ({})".format(scores.mean()*100, scores.std()*100))
```

```
Avg Score: 86.37090968418015% (0.2984484605833362)
```

In [44]:
```
alteredX = realX.copy()

xgb.fit(realX,realY)

real_pred = xgb.predict_proba(realX)[:,1]
for i in range(1000,4000,1000):
    alteredX['MonthlyIncome'] = realX['MonthlyIncome'] + i
    preds = xgb.predict_proba(alteredX)[:,1]
    print('Percent decrease in avg delinquincy rate with ${} in monthly inc
        -100*(preds.mean()-real_pred.mean())/real_pred.mean()))
```

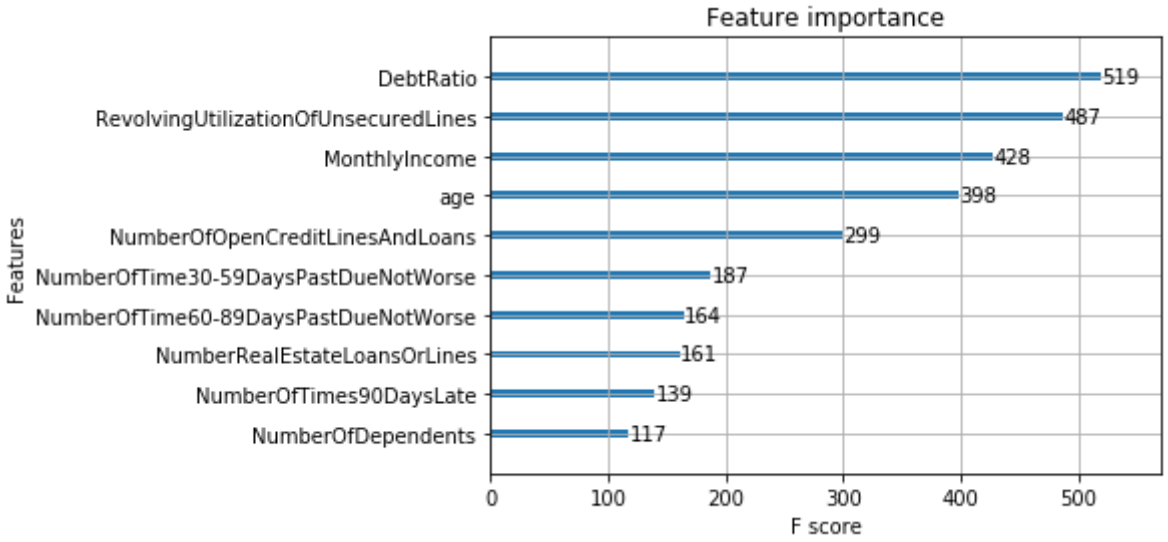
```
Percent decrease in avg delinquincy rate with $1000 in monthly income: 0.
9590820261309075%
Percent decrease in avg delinquincy rate with $2000 in monthly income: 2.
5288837980835255%
Percent decrease in avg delinquincy rate with $3000 in monthly income: 3.
9909223277206913%
```

With no other changes, increasing the monthly income of a sample will decrease the model predicting delinquincy. However, this does not account for how other features may change as a result of the change in income.

2) What is the most important variable in predicting deliquency? What is the most important pair of variables? Make a data science argument supported by data.

In [45]:
```
xgboost.plot_importance(xgb)
# xgb.feature_importances_
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x10f755a58>



In [46]:
```
from sklearn.feature_selection import SelectKBest,chi2
realX = realX.fillna(realX.mean())
for i in range(1,3):
    b = SelectKBest(chi2,i)
    b.fit(realX,realY)
    print('best {} features - chi score: {}'.format(i,realX.columns.get_val
    print('best {} features - roc_auc xgboost: {}'.format(i,realX.columns.g
```

```
best 1 features - chi score: ['MonthlyIncome']
best 1 features - roc_auc xgboost: ['DebtRatio']
best 2 features - chi score: ['NumberOfTimes90DaysLate' 'MonthlyIncome']
best 2 features - roc_auc xgboost: ['RevolvingUtilizationOfUnsecuredLine
s' 'DebtRatio']
```

Depending on the scoring function and the model, the best features are different, as listed above. According to the chi scorer, the best pair of features are MonthlyIncome and NumberOfTimes90DaysLate and the best single feature is MonthlyIncome

3) The Age Discrimination in Employment Act (ADEA) forbids age discrimination against people who are age 40 or older. Look at the best models you used in your Kaggle competion. Were they discriminating against older people? Make the best argument you can.

In [47]:
```
X = pd.concat([trainX,testX])
X1 = X[[a for a in X.columns if a != 'F5' and a != 'F19']]
Y1 = X['F5'][X['F5'].notnull()]
Y2 = X['F19'][X['F19'].notnull()]
f5train = X1[X['F5'].notnull()]
f19train = X1[X['F19'].notnull()]
f5test = X1[X['F5'].isnull()]
f19test = X1[X['F19'].isnull()]
enet = ElasticNetCV(alphas=[0.1,1,5,10],l1_ratio=[0.01,0.1,0.3,0.5,0.7,0.9]
enet.fit(f5train,Y1)
yf5 = enet.predict(f5test)
enet = ElasticNetCV(alphas=[0.1,1,5,10],l1_ratio=[0.01,0.1,0.3,0.5,0.7,0.9]
enet.fit(f19train,Y2)
yf19 = enet.predict(f19test)
```

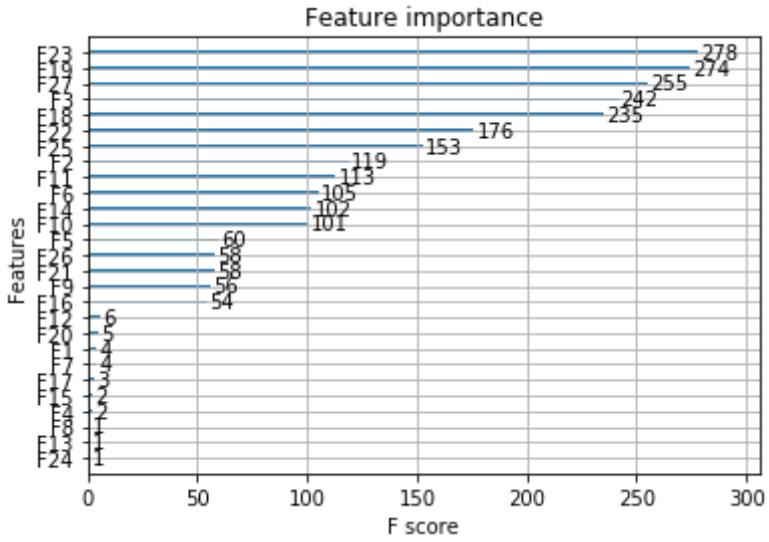
```
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/model_s
election/_split.py:1943: FutureWarning: You should specify a value for 'c
v' instead of relying on the default value. The default value will change
from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
  ConvergenceWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
  ConvergenceWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
```

In [48]:
```
Yt5 = X['F5'][X['F5'].isnull()]
df = pd.Series(yf5,index=Yt5.index)
X['F5'] = X['F5'].fillna(df)
Yt19 = X['F19'][X['F19'].isnull()]
df = pd.Series(yf19,index=Yt19.index)
X['F19'] = X['F19'].fillna(df)
```

In [49]:
```
trainX = X[:49998]
testX = X[49998:]
```

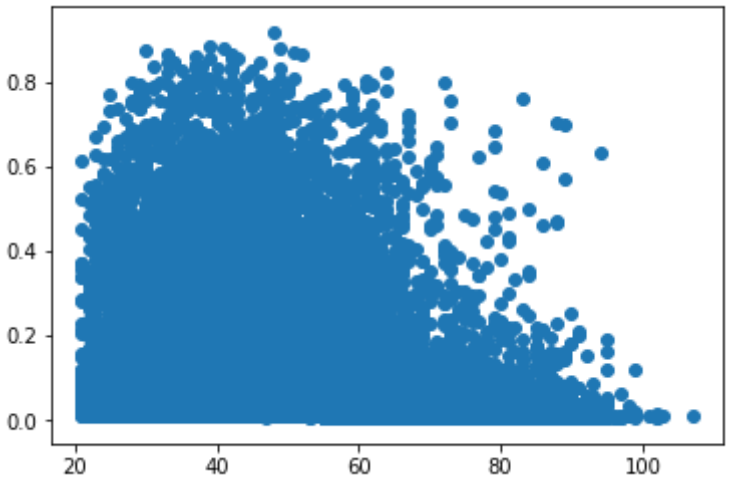
In [50]:
```
 = XGBClassifier(learning_rate=0.1, n_estimators= 200, max_depth= 6, min_ch
am = xgb.get_params()
train = xgboost.DMatrix(trainX.values,label=trainY.values)
 = xgboost.cv(param,xgbtrain,num_boost_round=param['n_estimators'],nfold=5,m
.set_params(n_estimators=cv.shape[0])
.fit(trainX,trainY,eval_metric='auc')
oost.plot_importance(xgb)
```

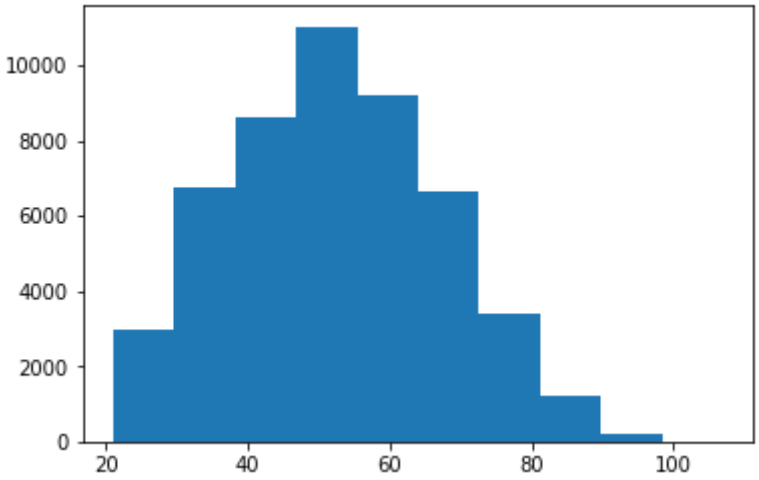
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x111c2e320>



Using our best submission, it seems as if age (F26) is not as important as any of other features (excluding number of dependents F5).

In [51]:
```
plt.scatter(trainX['F26'].values,xgb.predict_proba(trainX)[:,1])
plt.show()
plt.hist(trainX['F26'])
plt.show()
```





We can see that although most there are more predictions leaning towards delinquency in ages under 40, there are more people at low ages. The number of people over 40 are more sparse, but we see relatively the same distribution of predictions. Furthermore, it seems that people over 60 are less likely to be predicted as delinquents

4) Your manager asks if the number of dependents in the family (spouse, no of children) has an effect on loan deliquency. What does the data say? Calculate a p-value to express how confident you are.

In [52]:
```
realX = realData[realData.columns[1:]]
realY = realData[realData.columns[0]]
```

In [53]:
```
realData.groupby('NumberOfDependents')['SeriousDlqin2yrs'].mean()
```

Out[53]:
```
NumberOfDependents
0.0     0.058629
1.0     0.073529
2.0     0.081139
3.0     0.088263
4.0     0.103774
5.0     0.091153
6.0     0.151899
7.0     0.098039
8.0     0.083333
9.0     0.000000
10.0    0.000000
13.0    0.000000
20.0    0.000000
Name: SeriousDlqin2yrs, dtype: float64
```

In [54]:
```
from sklearn.feature_selection import SelectKBest,chi2
realX = realX.fillna(realX.mean())
b = SelectKBest(chi2,10)
b.fit(realX,realY)
print('p-value for effect of number of dependents on delinquency:',b.pvalue
```

```
p-value for effect of number of dependents on delinquency: 1.398057514620
496e-110
```

According to the data, the chance of delinquency increases as the number of dependents approaches 6, then decreases again. The p-value for the number of dependents is negligibly small, so we can safely reject the null hypothesis.

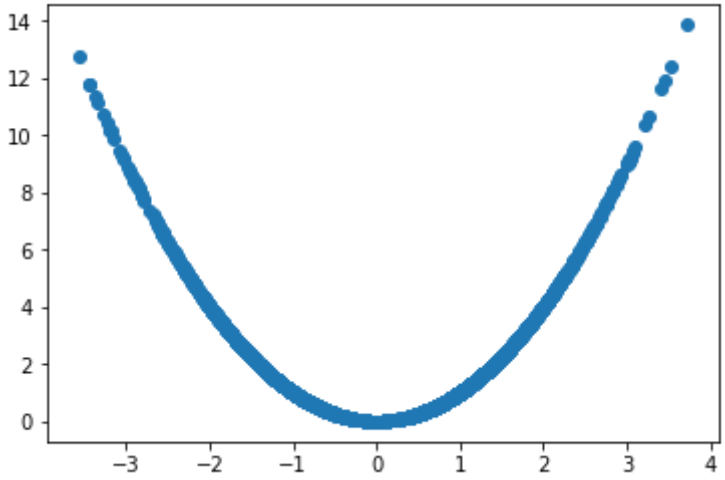
Problem 2.

b)

Create two continuous random variables X, Y so that X and Y are strongly dependent but the best linear regression fit y = β1x+β0 has the optimal β1 = 0. Show a scatter plot ofx, ypairs.

For the problem, we can use the example: in this case X is the a normal distrbution with E(X) = 0 and a Var(X) = 1, and have Y = X^2. These are uncorrelated variables but are also clearly depedent, as shown from the proof from written

In [3]:
```
n=10000
X = np.random.normal(size=n)
Y = X**2
plt.scatter(X,Y)
plt.show()
```



In [4]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(np.reshape(X,(n,1)),Y)
print('B1: ',linreg.coef_[0])
```

```
B1:  0.018186748359344623

/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/base.py:485: RuntimeWarning: internal gelsd driver lwork query erro
r, required iwork dimension not returned. This is likely the result of LA
PACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling ba
ck to 'gelss' driver.
  linalg.lstsq(X, y)
```

Thus, the with the equation y = β1 * x + β0, β1 is close to 0 as β1 represent the correlation between the two variables, thus creating the ideal linear regression.

In [ ]:
```
```