Load and analyze the data

First, lets load the data. Download the data from [UCI ML repository (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients)](https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients) and keep the excel sheet in the same directory as the notebook. Also you need to have the package `xlrd` installed on your machine.

In [1]:
```
import pandas as pd
card_data = pd.read_excel('default of credit card clients.xls', header = 1)
card_data.columns
```

Out[1]:
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AM
T2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default payment next month'],
      dtype='object')
```

In [2]:
```
card_data.head()
```

Out[2]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AM
0	1	20000	2	2	1	24	2	2	-1	-1	...	
1	2	120000	2	2	2	26	-1	2	0	0	...	3:
2	3	90000	2	2	2	34	0	0	0	0	...	14:
3	4	50000	2	2	1	37	0	0	0	0	...	28:
4	5	50000	1	2	1	57	-1	0	-1	0	...	20!

5 rows × 25 columns

Following are the Attribute Information taken from the UCI website:

This research employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. This study reviewed the literature and used the following 23 variables as explanatory variables:

- LIMIT_BAL: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- Sex (1 = male; 2 = female).
- Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- Marriage (1 = married; 2 = single; 3 = others).
- Age (year).
- PAY_0 to Pay_6: Repayment status of payment from Sep, Aug, Jul, Jun, May and Apr 2005 respectively

- BILL_AMT1 to BILL_AMT6: History of past payment from Sep, Aug, Jul, Jun, May and Apr 2005. The value -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- PAY_AMT1 to PAY_AMT6: Amount of previous payment (NT dollar) for months Sep, Aug, Jul, Jun, May and Apr 2005 respectively.
- default payment next month: Shows if the person defaulted in the following month, Oct 2005.

Next, lets look at how the distribution of 0 and 1 in the dataset and null values in any of the attributes

In [3]:
```
pd.isnull(card_data).any().any()
```

Out[3]:
```
False
```

In [4]:
```
card_data[['default payment next month', 'ID']].groupby('default payment ne
```

Out[4]:

	ID
default payment next month	
0	23364
1	6636

The data set's distribution is skewed towards the customers who dont default. For this reason when we perform cross validation we want the cross validation set to have the same distribution as the original and thus we will use `sklearn.model_selection.StratifiedKFold` cross validation. This ensures that the training data is split into the train and cross validation sets preserving the distribution of the original data set. Also there are no null values

---

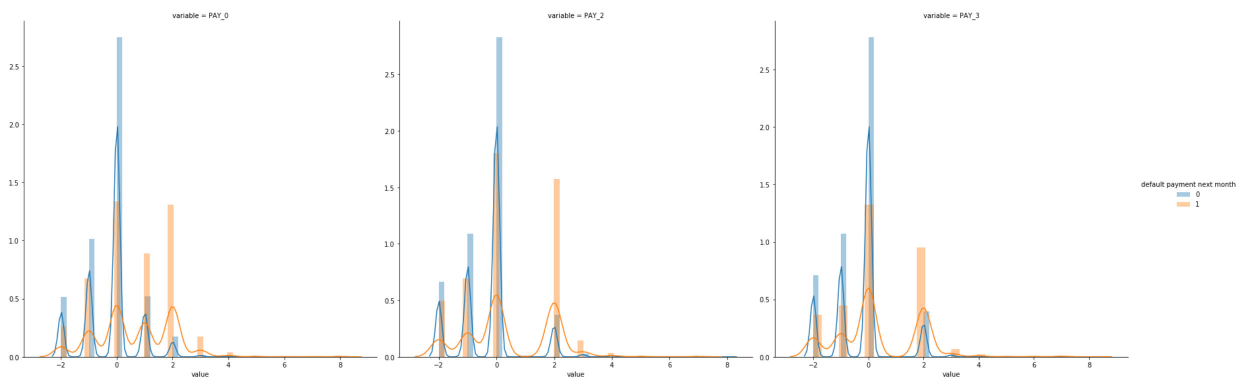
Lets look at how previous payment history relates to how the customer will default in next payment.

In [5]:
```
del card_data['ID']
```

In [6]:
```
m = pd.melt(card_data,  id_vars = ['default payment next month'],
        value_vars=['PAY_0', 'PAY_2', 'PAY_3'])
```

```
In [7]: %matplotlib inline
        import seaborn as sns
        g = sns.FacetGrid(m,
                          col = 'variable',
                          hue = 'default payment next month',
                          col_wrap = 3,
                          sharex = False,
                          sharey = False,
                          height = 8)
        g.map(sns.distplot, "value", kde = True).add_legend()
```

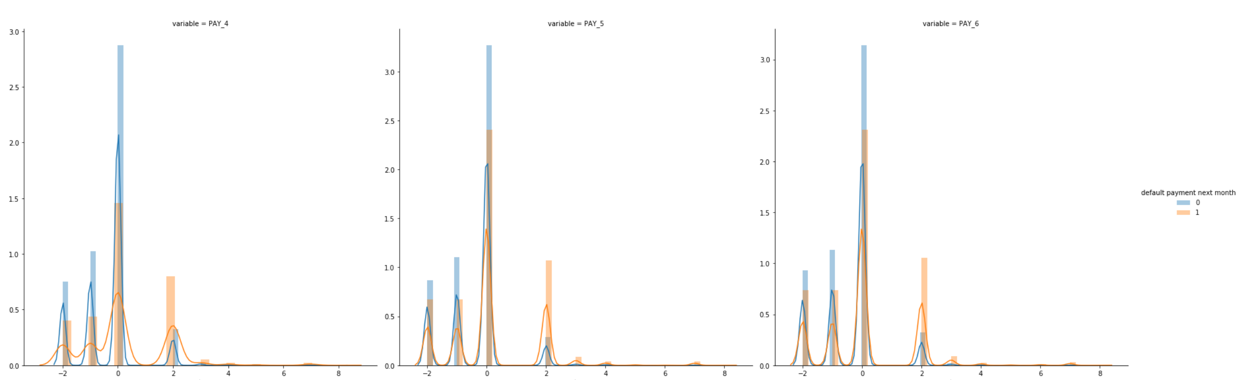
```
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1201f1668>
```




```
In [8]: m = pd.melt(card_data,  id_vars = ['default payment next month'],
                value_vars=['PAY_4', 'PAY_5', 'PAY_6'])
```

```
In [9]: %matplotlib inline
        import seaborn as sns
        g = sns.FacetGrid(m,
                          col = 'variable',
                          hue = 'default payment next month',
                          col_wrap = 3,
                          sharex = False,
                          sharey = False,
                          height = 8)
        g.map(sns.distplot, "value", kde = True).add_legend()
```

```
Out[9]: <seaborn.axisgrid.FacetGrid at 0x11e8522e8>
```



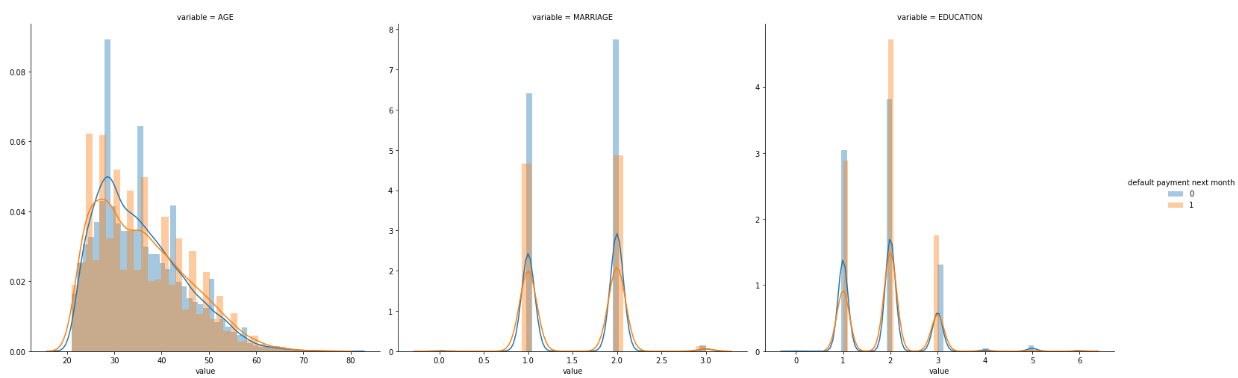
Clearly those who have made late payments in the past (ones with the value of PAY_X > 0) are more likely to default next and those who have made payments on or before time in the past are more likely to continue that trend. Thus it seems that these parameters are very likely a good estimator of who will default next.

Let's see how this relates to the education and age

In [10]:
```
m = pd.melt(card_data,  id_vars = ['default payment next month'],
         value_vars=['AGE', 'MARRIAGE', 'EDUCATION'])
```

In [11]:
```
%matplotlib inline
import seaborn as sns
g = sns.FacetGrid(m,
                  col = 'variable',
                  hue = 'default payment next month',
                  col_wrap = 3,
                  sharex = False,
                  sharey = False,
                  height = 7,
                  legend_out = True)
g.map(sns.distplot, "value", kde = True).add_legend()
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x11fe02400>



As we can see above people younger than approimately 25 and older than 41 are more likely to default and graduates are least likely to be defaulters. Marital status isnt a strong indicator though.

What we will do next is train three models each based on

- RandomForests
- LogisticRegression
- XGBoost's classifier

In [12]:
```python
def choose_best_model(models, X_train, Y_train, k = 10, random_state = 0):
    # Given a list of models X_train and Y_train, the function does stratif
    # models and returns the on with best average accuracy. The function re
    #
    #   Model with best cross validation accuracy
    #   Cross validation accuracy
    best_model, best_accuracy = None, 0
    from sklearn.model_selection import StratifiedKFold
    from sklearn.metrics import accuracy_score
    for i, model in enumerate(models):
        print('Running', k, '- fold cross validation on model', (i + 1), 'c
        model_accuracy = []
        kf = StratifiedKFold(n_splits = k, random_state = random_state)
        for train_index, test_index in kf.split(X_train, Y_train):
            model.fit(X_train.iloc[train_index, :], Y_train.iloc[train_inde
            model_accuracy.append(
                accuracy_score(Y_train.iloc[test_index], model.predict(X_tr

        mean_accuracy = sum(model_accuracy) / len(model_accuracy)
        if best_accuracy < mean_accuracy:
            best_accuracy = mean_accuracy
            best_model = model

    return best_model, best_accuracy
```

In [13]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(card_data[['LIMIT_BAL',
       'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']], card_d
                 train_size = 0.8)
```

In [14]:
```python
print('Percentage of 1s in train and test set is,', Y_train.sum() / Y_trair
      'and', Y_test.sum() / Y_test.shape[0], 'respectively')
```

```
Percentage of 1s in train and test set is, 0.22304166666666667 and 0.2138
3333333333332 respectively
```

In [15]:
```python
from sklearn.ensemble import RandomForestClassifier
rf_models = [RandomForestClassifier(n_estimators = n) for n in [100, 150, 2
best_rf_model, rf_crossval_accuracy = choose_best_model(rf_models, X_train,
```

```
Running 10 - fold cross validation on model 1 of 7 models
Running 10 - fold cross validation on model 2 of 7 models
Running 10 - fold cross validation on model 3 of 7 models
Running 10 - fold cross validation on model 4 of 7 models
Running 10 - fold cross validation on model 5 of 7 models
Running 10 - fold cross validation on model 6 of 7 models
Running 10 - fold cross validation on model 7 of 7 models
```

In [16]:
```
from sklearn.metrics import accuracy_score
train_accuracy, test_accuracy = \
accuracy_score(Y_train, best_rf_model.predict(X_train)), accuracy_score(Y_t
print('The chosen random forest model gave a training accuracy, cross val a
      train_accuracy, rf_crossval_accuracy, test_accuracy, 'respectively')
print('Best model has', best_rf_model.n_estimators, 'estimators')
```

```
The chosen random forest model gave a training accuracy, cross val accura
cy and test_accuracy of 0.9819166666666667 0.817917932349757 0.8181666666
666667 respectively
Best model has 300 estimators
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
lr_models = [LogisticRegression(max_iter = n, solver = 'lbfgs') for n in [5
best_lr_model, lr_crossval_accuracy = choose_best_model(lr_models, X_train,
```

```
Running 10 - fold cross validation on model 1 of 5 models
Running 10 - fold cross validation on model 2 of 5 models
Running 10 - fold cross validation on model 3 of 5 models
Running 10 - fold cross validation on model 4 of 5 models
Running 10 - fold cross validation on model 5 of 5 models
```

In [18]:
```
train_accuracy, test_accuracy = \
accuracy_score(Y_train, best_lr_model.predict(X_train)), accuracy_score(Y_t
print('The chosen Logistic Regression model gave a training accuracy, cross
      train_accuracy, rf_crossval_accuracy, test_accuracy, 'respectively')
print('Best model has', best_lr_model.max_iter, 'iterations')
```

```
The chosen Logistic Regression model gave a training accuracy, cross val
accuracy and test_accuracy of 0.7769583333333333 0.817917932349757 0.786
respectively
Best model has 500 iterations
```

In [19]:
```
from xgboost import XGBClassifier
xgb_models = [XGBClassifier(n_estimators=n) for n in [100, 150, 200, 250, 3
best_xgb_model, xgb_crossval_accuracy = choose_best_model(xgb_models, X_tra
```

```
Running 10 - fold cross validation on model 1 of 6 models
Running 10 - fold cross validation on model 2 of 6 models
Running 10 - fold cross validation on model 3 of 6 models
Running 10 - fold cross validation on model 4 of 6 models
Running 10 - fold cross validation on model 5 of 6 models
Running 10 - fold cross validation on model 6 of 6 models
```

In [20]:
```
train_accuracy, test_accuracy = \
accuracy_score(Y_train, best_xgb_model.predict(X_train)), accuracy_score(Y_
print('The chosen XGBoost classifier model gave a training accuracy, cross
      train_accuracy, xgb_crossval_accuracy, test_accuracy, 'respectively')
print('Best model has', best_xgb_model.n_estimators, 'estimators')
```

```
The chosen XGBoost classifier model gave a training accuracy, cross val a
ccuracy and test_accuracy of 0.8255 0.8220012662328587 0.8226666666666667
respectively
Best model has 100 estimators
```

In [30]:

```
from sklearn.metrics import confusion_matrix

rf_pred = best_rf_model.predict(X_test)
rf_confusion_matrix = confusion_matrix(Y_test, rf_pred)
print('-' * 20)
print('Confusion Matrix for RandomForestClassifier')
print(rf_confusion_matrix)

lr_pred = best_lr_model.predict(X_test)
lr_confusion_matrix = confusion_matrix(Y_test, lr_pred)
print('-' * 20)
print('Confusion Matrix for LogisticRegression is')
print(lr_confusion_matrix)

xgb_pred = best_xgb_model.predict(X_test)
xgb_confusion_matrix = confusion_matrix(Y_test, xgb_pred)
print('-' * 20)
print('Confusion Matrix for XGBoostClassification is')
print(xgb_confusion_matrix)
print('-' * 20)
```

```
--------------------
Confusion Matrix for RandomForestClassifier
[[4460  257]
 [ 834  449]]
--------------------
Confusion Matrix for LogisticRegression is
[[4716    1]
 [1283    0]]
--------------------
Confusion Matrix for XGBoostClassification is
[[4505  212]
 [ 852  431]]
--------------------
```

In [31]:
```
m sklearn.metrics import recall_score
m sklearn.metrics import precision_score
m sklearn.metrics import f1_score


prec, rf_recall, rf_f1 = precision_score(Y_test, rf_pred), recall_score(Y_te
nt('-' * 80)
nt('Precision, Recall and F1 score of RandomForest is', rf_prec, rf_recall,

prec, lr_recall, lr_f1 = precision_score(Y_test, lr_pred), recall_score(Y_te
nt('-' * 80)
nt('Precision, Recall and F1 score of LogisticRegression is', lr_prec, lr_re

_prec, xgb_recall, xgb_f1 = precision_score(Y_test, xgb_pred), recall_score(
nt('-' * 80)
nt('Precision, Recall and F1 score of XGBoost is', xgb_prec, xgb_recall, xgb
```

```
--------------------------------------------------------------------------
-------
Precision, Recall and F1 score of RandomForest is 0.6359773371104815 0.34
996102883865937 0.45148315736551026 respectively
--------------------------------------------------------------------------
-------
Precision, Recall and F1 score of LogisticRegression is 0.0 0.0 0.0 respe
ctively
--------------------------------------------------------------------------
-------
Precision, Recall and F1 score of XGBoost is 0.6702954898911353 0.3359314
1075604055 0.44755970924195226 respectively
```

Conclusion

The accuracy score of both `RandomForestClassifier` and `XGBClassifier` are around 81% which is a reasonable start given the limited amount of data available. Its not clear on which model is better, we will however choose `XGBClassifier` as it has slightly better precision and it generalized well on all three data sets.

In [ ]: