7 Feature Selection

We now import standard packages and functions calling the `dcr` module using `from dcr import *` and ignore warning messages.

```
import warnings; warnings.simplefilter('ignore')
from dcr import *
```

We run `%matplotlib inline` to see the full outputs, and specify the resolution of the figures.

```
%matplotlib inline
plt.rcParams['figure.dpi'] = 300
plt.rcParams['figure.figsize'] = (16, 9)
plt.rcParams.update({'font.size': 16})
```

7.1 Synopsis

In this chapter, we show some efficient techniques for feature selection, which are important to describe default events. We separate techniques into economic, univariate and model-based feature selection. Economic feature selection is very popular as it aims to identify the data generating processes. Proponents of this technique argue that theory-based models are more robust in the long run. Stand-alone and model-based feature selection relies on the identification of best fitting variables either in an univariate or multi-variate fashion.

The presented feature selection techniques may be applied for selecting features for other outcome variables like payoffs, loss rates given default and exposures.

7.2 Economic Feature Selection

Economic feature selection is very popular as it aims to identify the data generating (i.e., true) processes. The vibrant research in academia and industry on the drivers of default and modeling concepts include:

- Idiosyncratic factors: features for borrowers, lender, guarantors and collateral;
- Time-varying externalities: macroeconomy;
- Time-constant externalities: market infrastructure and regulations.

For idiosyncratic factors, we may consider borrower features such as income, wealth and liquidity. For time-varying externalities, we may consider macro-economic factors. There is an ongoing discussion as to the degree of macroeconomic information that can be included in models and we will elaborate on this in our development of through-the-cycle and point-in-time models in the default modeling chapter. For time-constant externalities we may consider market infrastructure, regional laws and regulations.

As we require the codes of this section throughout the book, we have summarized these in the function `dataprep()`, which we can call as follows:


```
from dcr import *
data, data_train, data_test, X_train_scaled, X_test_scaled, y_train, y_test = dataprep(
    data, depvar='default_time', splitvar='time', threshold=26)
```

The resulting dataset `data` is the complete data. Subsets for the various applications in subsequent chapter are `data_train`, `data_test`, `X_train_scaled`, `X_test_scaled`, `y_train`, and `y_test`. The function `dataprep` includes all codes explained in this sections and we summarize thereafter.

7.2.1 Liquidity and Equity

Liquidity and equity are important features. For mortgage risk models, the "double-trigger" theory assumes that mortgage borrower default requires two triggers: a liquidity or equity constraint. A liquidity constraint is the inability of a borrower to make scheduled loan payments and an equity constraint means that the value of the assets is below the value of outstanding debt, which is also known as negative equity. In other words, liquidity and equity constraints are joint drivers of default. Asset values for mortgages may include collateral in the case of non-recourse mortgage lending and collateral plus general borrower assets in the case of recourse lending.

Let's look at liquidity first. Borrower liquidity may change due to changes in income (e.g., salary), expenses (e.g., for food, housing and transport) and access to external liquidity (e.g., by refinance or second lien loans).

We approximate borrower liquidity by cumulative excess payments (CEP) `cep_time`. CEP is the scheduled balance minus the actual balance over the property value to control for size. Mortgage Borrowers make generally annuity payments that can be computed from the original loan balance `balance_orig_time`, mortgage interest rate `interest_rate_time`, and the maturity at loan origination (i.e., `mat_time` minus `orig_time`).

```
data.loc[:, 'annuity'] = ((data.loc[:,'interest_rate_time']/(100*4)) * data.loc[:,'
    balance_orig_time']) / (1-(1+data.loc[:,'interest_rate_time']/(100*4))**(-(data.loc[:,'
    mat_time'] - data.loc[:,'orig_time'])))
```

The scheduled balance is computed based on the original loan balance `balance_orig_time`, periodic annuity payments `annuity`, mortgage interest rate `interest_rate_time`, and the maturity at loan origination (i.e., `mat_time` minus `orig_time`).

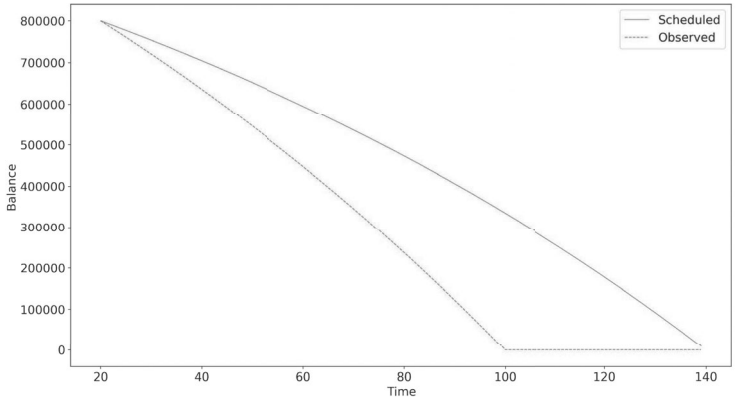
```
data.loc[:,'balance_scheduled_time']  = data.loc[:,'balance_orig_time'] * (1+data.loc[:,'
    interest_rate_time']/(100*4))**(data.loc[:,'time'] - data.loc[:,'orig_time']) - data.
    loc[:,'annuity'] * ((1+data.loc[:,'interest_rate_time']/(100*4))**(data.loc[:,'time']-
    data.loc[:,'orig_time'])-1) / (data.loc[:,'interest_rate_time']/(100*4))
```

Let's look at an example where a loan is originated in period 20 and matures in period 140 (i.e., after 30 years/120 quarters). The loan amount is $500,000 and the interest rate is 2.5%. We benchmark the scheduled balance with the observed balance if the borrower repays the loan in period 100 (i.e. after 20 years/80 quarters) in annuities.

We generate a `pandas` dataframe for time and generate the additional variables, compute the scheduled and observed annuity and balance and plot the balances.

```
time = np.arange(20,140,1)
example=pd.DataFrame(data=time, index=time, columns=["time"])

example.loc[:, 'orig_time'] = 20
example.loc[:, 'mat_time'] = 140
```

```
example.loc[:, 'mat_time_observed'] = 100
example.loc[:, 'balance_orig_time'] = 800000
example.loc[:, 'interest_rate_time'] = 2.5

example.loc[:, 'annuity'] = ((example.loc[:,'interest_rate_time']/(100*4)) * example.loc[:,
    'balance_orig_time']) / (1-(1+example.loc[:,'interest_rate_time']/(100*4))**(-(example.
    loc[:,'mat_time'] - example.loc[:,'orig_time'])))
example.loc[:, 'annuity_observed'] = ((example.loc[:,'interest_rate_time']/(100*4)) *
    example.loc[:,'balance_orig_time']) / (1-(1+example.loc[:,'interest_rate_time']/(100*4)
    )**(-(example.loc[:,'mat_time_observed'] - example.loc[:,'orig_time'])))

example.loc[:,'balance_scheduled_time']  = example.loc[:,'balance_orig_time'] * (1+example.
    loc[:,'interest_rate_time']/(100*4))**(example.loc[:,'time'] - example.loc[:,'orig_time
    ']) - example.loc[:,'annuity'] * ((1+example.loc[:,'interest_rate_time']/(100*4))**(
    example.loc[:,'time']-example.loc[:,'orig_time'])-1) / (example.loc[:,'
    interest_rate_time']/(100*4))
example.loc[:,'balance_time']  = example.loc[:,'balance_orig_time'] * (1+example.loc[:,'
    interest_rate_time']/(100*4))**(example.loc[:,'time'] - example.loc[:,'orig_time']) -
    example.loc[:,'annuity_observed'] * ((1+example.loc[:,'interest_rate_time']/(100*4))**(
    example.loc[:,'time']-example.loc[:,'orig_time'])-1) / (example.loc[:,'
    interest_rate_time']/(100*4))
example = example.clip(lower=0)

plt.plot('time', 'balance_scheduled_time', data=example, linewidth=1, label='Scheduled')
plt.plot('time', 'balance_time', data=example, color='red', linewidth=1, label='Observed',
    linestyle='dashed')
plt.xlabel('Time')
plt.ylabel('Balance')
plt.legend(loc='best')
plt.show()
```

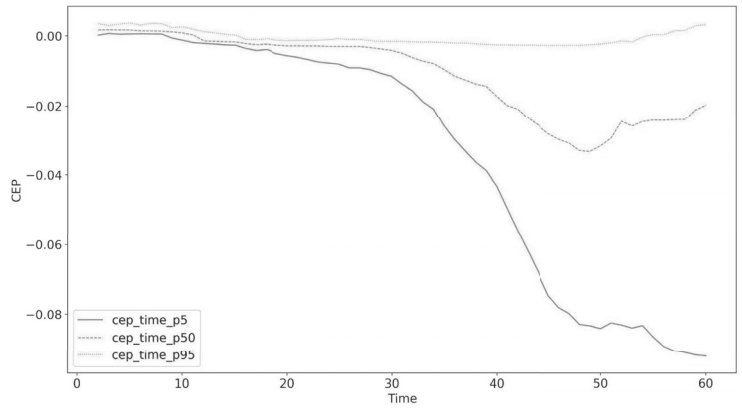
We then compare the scheduled balance with the observed balance `balance_time`. A positive (negative) difference implies that borrowers repay at a greater (lower) speed than scheduled.

We now revert to our mortgage data and calculate a liquidity ratio `cep_time` by scaling this difference by the property value at origination `property_orig_time` which we infer from the loan amount and LTV ratio at loan origination. We do not use dynamic property values as these changes overlap with the equity variable that we will generate next if both are included in the same model.

```
data.loc[:,'property_orig_time'] = np.maximum(data.loc[:,'balance_orig_time'] / (data.loc
    [:,'LTV_orig_time']/100),100000)

data.loc[:,'cep_time']= (data.loc[:,'balance_scheduled_time'] - data.loc[:,'balance_time'])
     / data.loc[:,'property_orig_time']
```

Let's now analyze the mean liquidity ratio `cep_time` as well as its lower and upper quartile over time:

```
cep_time_p25 = data.groupby('time')['cep_time'].quantile(0.25).reset_index(drop=False)
cep_time_p50 = data.groupby('time')['cep_time'].median().reset_index(drop=False)
cep_time_p75 = data.groupby('time')['cep_time'].quantile(0.75).reset_index(drop=False)

plt.plot('time', 'cep_time', data=cep_time_p25, label='cep_time_p5')
plt.plot('time', 'cep_time', data=cep_time_p50, color='red', linewidth=1, label='
    cep_time_p50',linestyle='dashed')
plt.plot('time', 'cep_time', data=cep_time_p75, color='green', linewidth=1, label='
    cep_time_p95',linestyle='dotted')
plt.xlabel('Time')
plt.ylabel('CEP')
plt.legend(loc='best')
plt.show()
```



It is apparent that during the financial crisis (i.e., from period 27) liquidity decreases. Note that there is also cross-sectional variation, which is visualized by the lower and upper quartile. Notably, the dispersion of liquidity has increased post-crisis.

Let's look at equity next. Borrower equity is generally based on current assets less liabilities. Often we assume that borrowers own a single asset (i.e., the house) funded by a single liability (the mortgage). Asset values may change due to changes in supply and demand in the housing markets, monetary policy or bank lending.

We approximate borrower equity by the home equity, which is backed out from the current loan balance `balance_time` and the current LTV `LTV_time`. A positive (negative) value implies that house value exceeds (is below) the outstanding debt value.

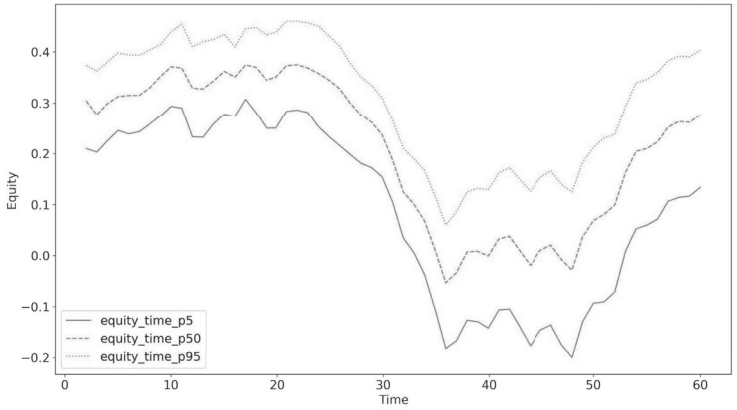
```
data.loc[:,'equity_time'] = 1-(data.loc[:,'LTV_time']/100)
```

We analyze the mean equity ratio `equity_time` as well as its lower and upper quartile over time:

```
equity_time_p25 = data.groupby('time')['equity_time'].quantile(0.25).reset_index(drop=False
    )
equity_time_p50 = data.groupby('time')['equity_time'].mean().reset_index(drop=False)
equity_time_p75 = data.groupby('time')['equity_time'].quantile(0.75).reset_index(drop=False
    )

plt.plot('time', 'equity_time', data=equity_time_p25, label='equity_time_p5')
plt.plot('time', 'equity_time', data=equity_time_p50, color='red', label='equity_time_p50',
    linestyle='dashed')
plt.plot('time', 'equity_time', data=equity_time_p75, color='green', label='equity_time_p95
    ',linestyle='dotted')
plt.xlabel('Time')
plt.ylabel('Equity')
```



```
plt.legend(loc='best')
plt.show()
```



Equity also decreases during the financial crisis (i.e., from period 27) and subsequently recovers from period 50 onwards.

Liquidity and equity are highly positively correlated as they are driven by similar features of the economy and the borrower:

```
data2=data[['cep_time', 'equity_time']]

print(data2.corr(method='pearson').round(decimals=2))
```

```
             cep_time  equity_time
cep_time         1.00         0.61
equity_time      0.61         1.00
```

We now analyze the path of a defaulting borrower. As liquidity and equity change, so does the default intensity (e.g., default rate). For example, for defaulted borrower with ID 9 (`data.id == 9`) we observe declining liquidity and equity levels on the path to default:

```
data2=data.loc[(data.id == 9),['time', 'cep_time', 'equity_time']]

plt.plot('time', 'cep_time', data=data2, label='cep_time')
plt.plot('time', 'equity_time', data=data2, color='red', label='equity_time',linestyle='
    dashed')
plt.xlabel('Time')
plt.ylabel('CEP and Equity')
plt.legend(loc='best')
```