总结报告 李木晗

I. Comprehensive Data Exploration

1. check the decoration

```
train.columns
```

```
Index(['id', '性别', '年龄', '体检日期', '*天门冬氨酸氨基转换酶', '*丙氨酸氨基转换酶',
       '*碱性磷酸酶', '*r-谷氨酰基转换酶', '*总蛋白', '白蛋白', '*球蛋白', '白球比例',
       '甘油三酯', '总胆固醇', '高密度脂蛋白胆固醇', '低密度脂蛋白胆固醇', '尿素', '肌酐',
       '尿酸', '乙肝表面抗原', '乙肝表面抗体', '乙肝e抗原', '乙肝e抗体', '乙肝核心抗体',
       '白细胞计数', '红细胞计数', '血红蛋白', '红细胞压积', '红细胞平均体积',
       '红细胞平均血红蛋白量', '红细胞平均血红蛋白浓度', '红细胞体积分布宽度', '血小板计数',
       '血小板平均体积', '血小板体积分布宽度', '血小板比积', '中性粒细胞%', '淋巴细胞%',
       '单核细胞%', '嗜酸细胞%', '嗜碱细胞%', '血糖'],
      dtype='object')
```

2. descriptive statistics summary

```
train['血糖'].describe()
```

```
count    5642.000000
mean        5.631925
std         1.544882
min         3.070000
25%         4.920000
50%         5.290000
75%         5.767500
max        38.430000
Name: 血糖, dtype: float64
```

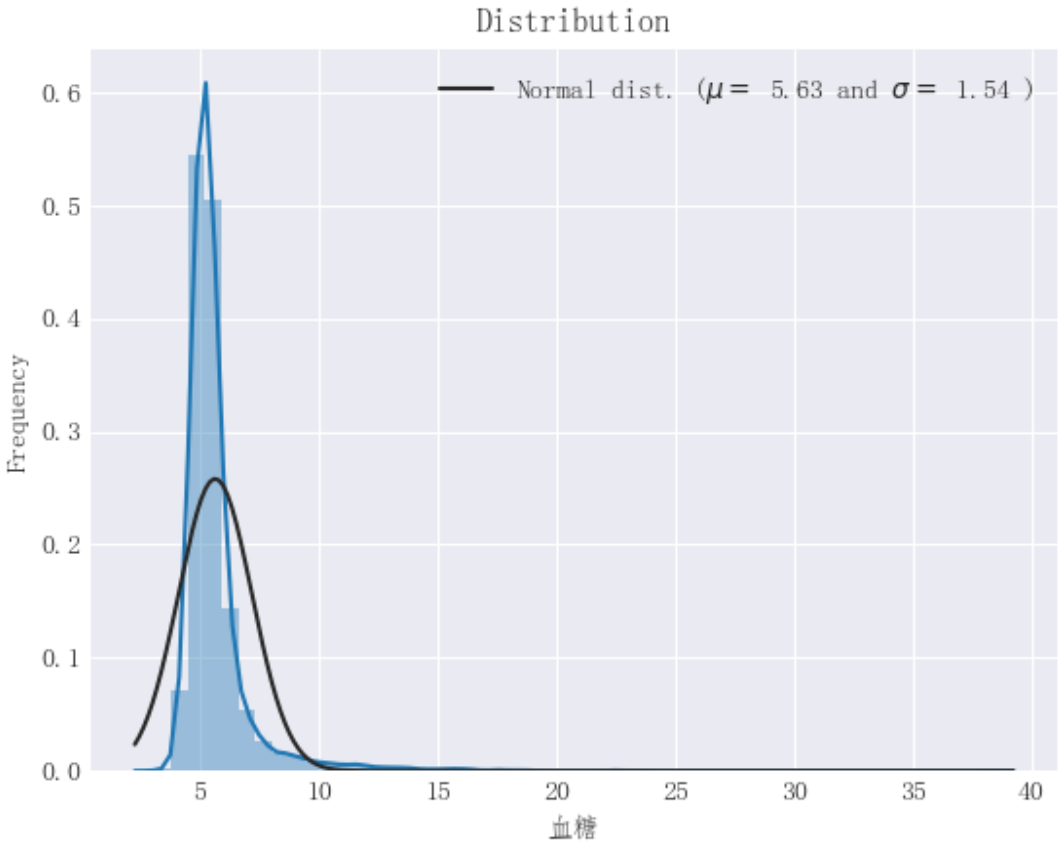
3. histogram

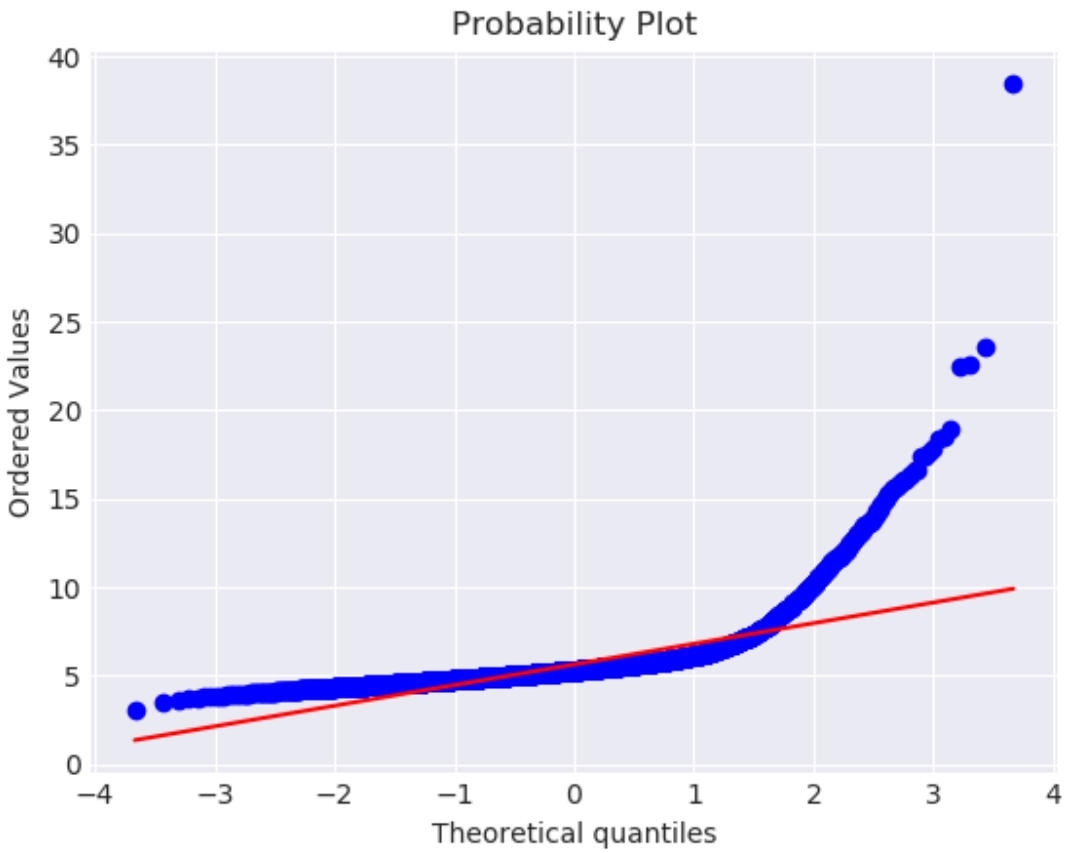
```
sns.distplot(train[item_to_do], fit=norm)

# Get the fitted parameters used by the function
(mu, sigma) = norm.fit(train[item_to_do])
print('\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))

# Now plot the distribution
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
            loc='best')
plt.ylabel('Frequency')
plt.title('Distribution')
```

```
# Get also the QQ-plot
fig = plt.figure()
res = stats.probplot(train[item_to_do], plot=plt)
plt.show()
```





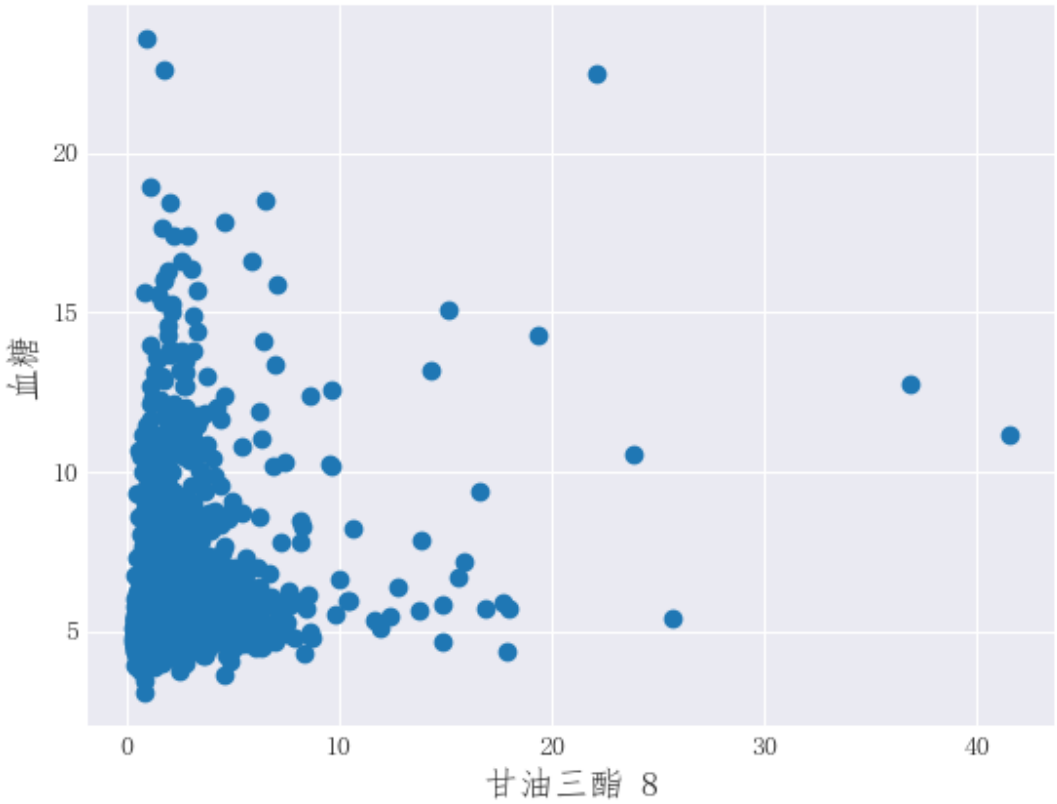
4. skewness and kurtosis

```
print("Skewness: %f" % train['血糖'].skew())
print("Kurtosis: %f" % train['血糖'].kurt())
```

```
Skewness: 5.551989
Kurtosis: 59.163792
```

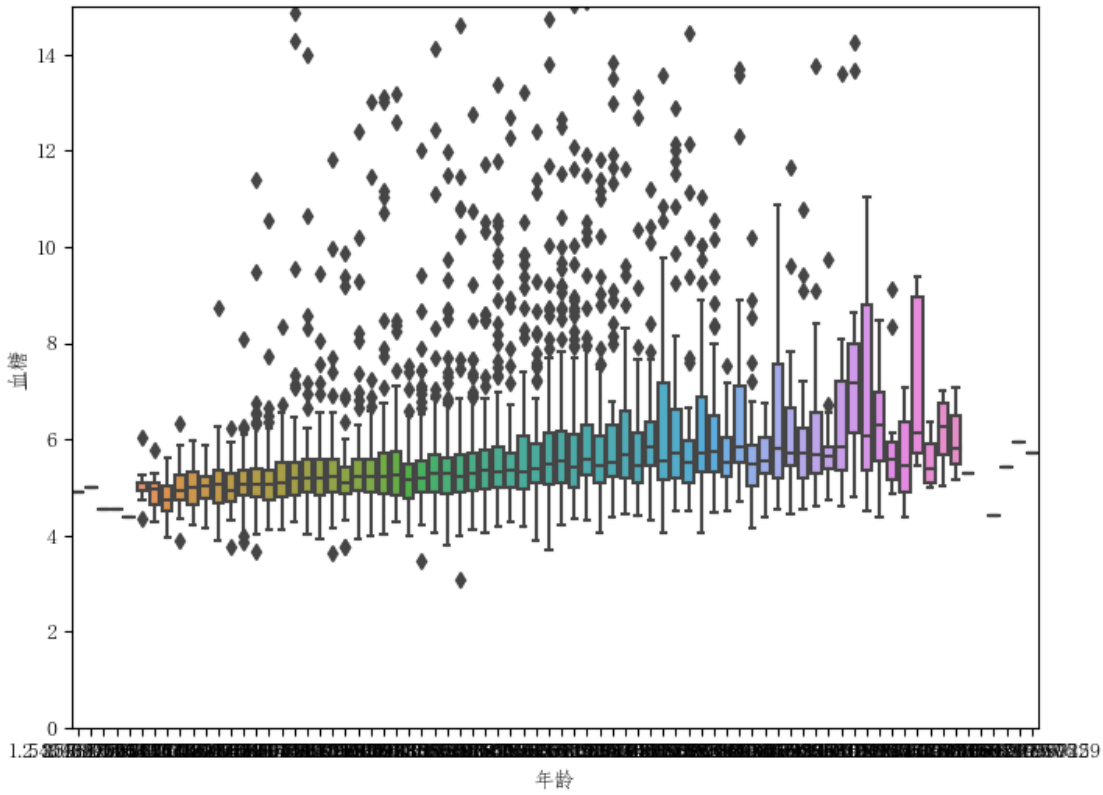
5. scatter plot, *Relationship with numerical variables*

```
i = 0
plt.rcParams['font.sans-serif'] = ['FangSong']
for item in test_item:
    fig, ax = plt.subplots()
    ax.scatter(x = train[item], y = train[target_item])
    plt.ylabel(target_item, fontsize=13)
    plt.xlabel(item + f' {i}', fontsize=13)
    name = item
    if item[0] == '*':
        name = name[1:]
    name = str(i) + ' ' + name
    i += 1
    plt.show()
```



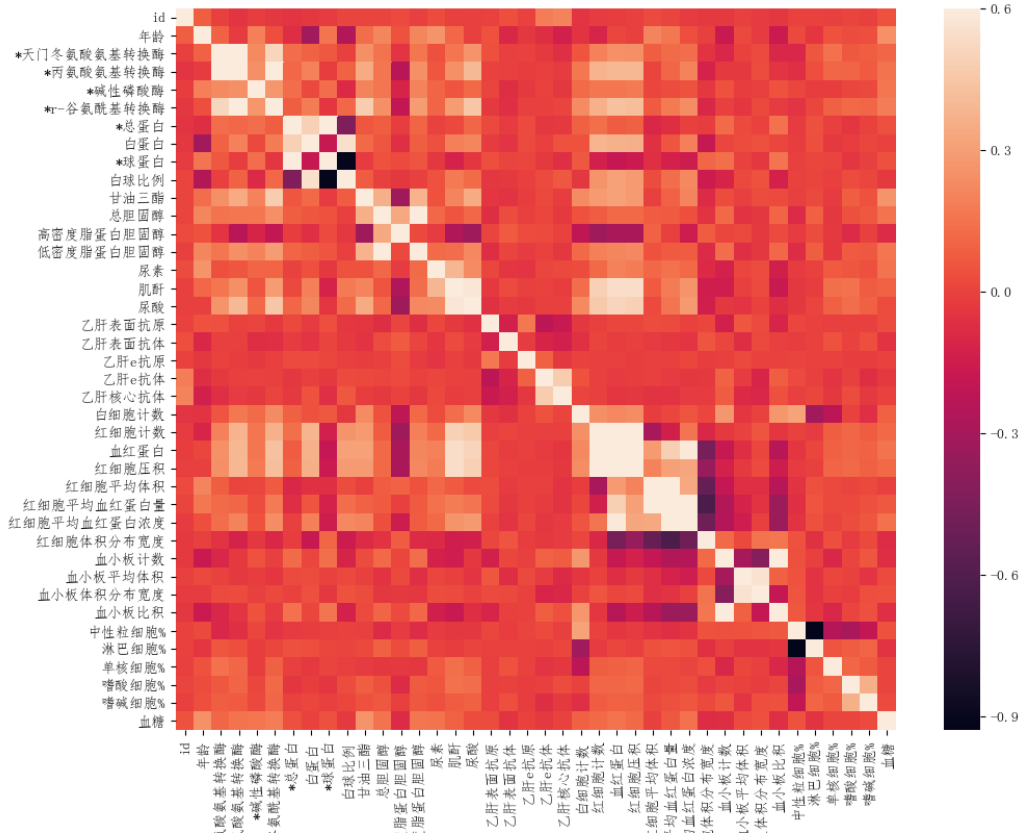
6. box plot, *Relationship with categorical variables*

```
var = '年龄'
data = pd.concat([train['血糖'], train[var]], axis=1)
plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y='血糖', data=data)
fig.axis(ymin=0, ymax=15)
plt.show()
```

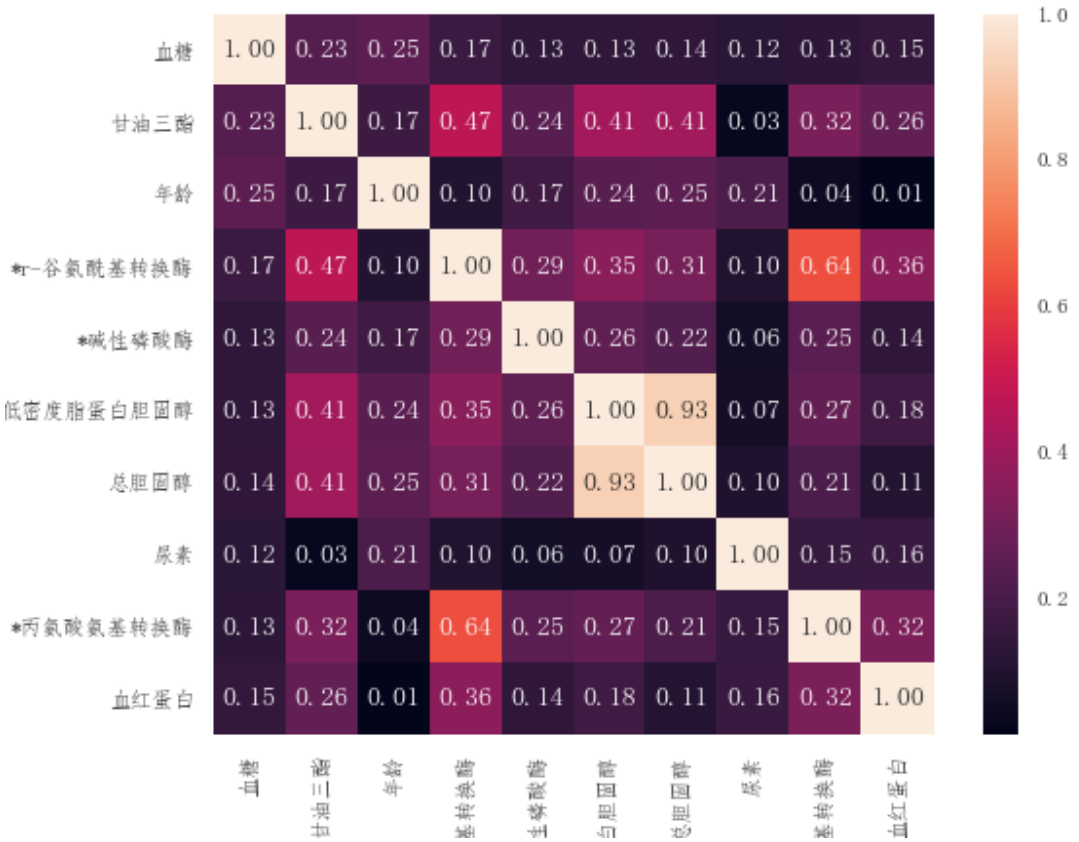


7. correlation heat-map

```
corrmat = train.corr()
plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=0.6, square=True)
plt.show()
```

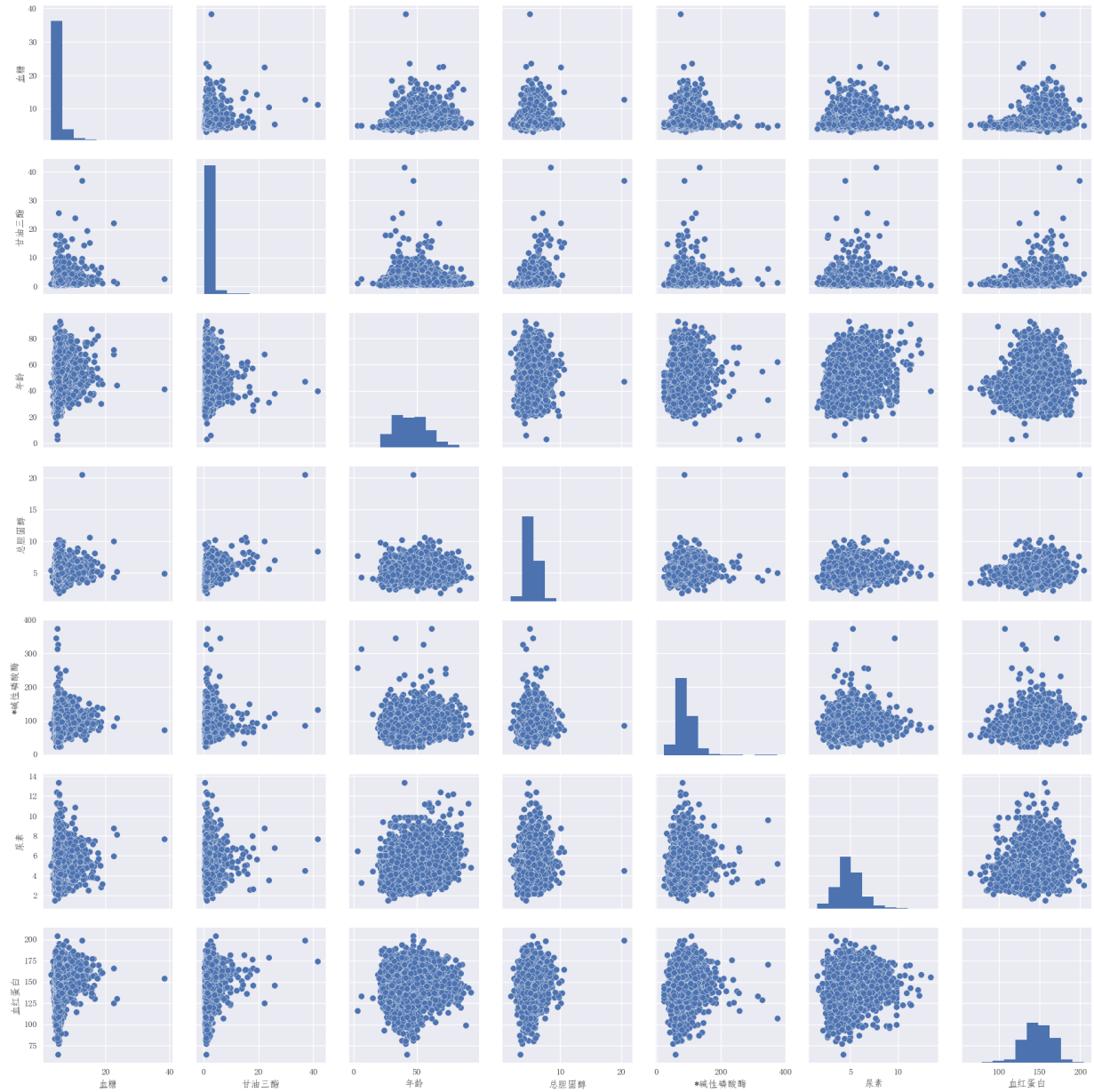



```
k = 10  # number of variables for heatmap
cols = corrmat.nlargest(k, '血糖')['血糖'].index
cm = np.corrcoef(train[cols].values.T)
sns.set(font_scale=0.8)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',
                 annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
plt.show()
```



8. scatter plots

```
cols = ['血糖', '甘油三酯', '年龄', '总胆固醇', '*碱性磷酸酶', '尿素', '血红蛋白']
sns.pairplot(train[cols], size=2.5)
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
plt.show()
```



9. missing data

```
total = train.isnull().sum().sort_values(ascending=False)
percent = (train.isnull().sum() / train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
print(missing_data)
```

```
              Total   Percent
乙肝表面抗体        4279  0.758419
乙肝核心抗体        4279  0.758419
乙肝表面抗原        4279  0.758419
乙肝e抗原          4279  0.758419
乙肝e抗体          4279  0.758419
尿酸             1378  0.244240
尿素             1378  0.244240
肌酐             1378  0.244240

*r-谷氨酰基转换酶     1221  0.216413
```

```
白球比例            1221  0.216413
*球蛋白            1221  0.216413
*总蛋白            1221  0.216413
白蛋白             1221  0.216413
*碱性磷酸酶          1221  0.216413
*丙氨酸氨基转换酶      1221  0.216413
*天门冬氨酸氨基转换酶    1221  0.216413
甘油三酯            1219  0.216058
总胆固醇            1219  0.216058
高密度脂蛋白胆固醇      1219  0.216058
低密度脂蛋白胆固醇      1219  0.216058
血小板平均体积          23  0.004077
血小板体积分布宽度        23  0.004077
血小板比积             23  0.004077
中性粒细胞%           16  0.002836
嗜酸细胞%            16  0.002836
嗜碱细胞%            16  0.002836
单核细胞%            16  0.002836
淋巴细胞%            16  0.002836
白细胞计数            16  0.002836
红细胞计数            16  0.002836
血红蛋白             16  0.002836
红细胞压积            16  0.002836
红细胞平均体积          16  0.002836
红细胞平均血红蛋白量       16  0.002836
红细胞平均血红蛋白浓度      16  0.002836
红细胞体积分布宽度        16  0.002836
血小板计数            16  0.002836
年龄                0  0.000000
性别                0  0.000000
血糖                0  0.000000
体检日期              0  0.000000
id                0  0.000000
```

这就是个对数据的brief overview吧，对这组数据来说、、普通的右偏正态（然而貌似并不是比较正常的**对数正态分布**[*作log变换使其变为正态*]），然而相关性不强，特征不明显，缺省值太多。。然后就折腾特征吧

II. Features engineering

1. **缺省值**
    - 缺失比例高的，可以舍去
    - 正态数据使用Mean描述
    - 非正态数据使用Median描述
    - 其他的填充方法，如SVD、回归预测
2. **标准化、归一化**

    特别是对线性模型，修正数据使得它符合线性性要求：

    对响应变量或者预测变量或者对于两者同时使用合适的非线性变换。

- linear function conversion

```
for item in test_item:
    maximum = max(test[item].max(), train[item].max())
    minimum = min(test[item].min(), train[item].min())
    test[item] = (test[item] - minimum) / (maximum - minimum)
    train[item] = (train[item] - minimum) / (maximum - minimum)
```

- box-cox transformation

  The Box-Cox transform is given by:

```
y = (x**lmbda - 1) / lmbda,  for lmbda > 0
    log(x),                  for lmbda = 0
```

  Thus we have:

```
def inverse_box_cox(y, ld):
    if ld == 0:
        return np.exp(y)
    else:
        return np.exp(np.log(ld * y + 1) / ld)
```

3. One Hot Encoding

   One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Problem with label encoding is that it assumes higher the categorical value, better the category.

4. 构造新特征

   - GBDT
   - CNN
   - 运算

```
all_data[f'{a}+{b}'] = all_data[a] + all_data[b]
all_data[f'{a}-{b}'] = all_data[a] - all_data[b]
all_data[f'{a}*{b}'] = all_data[a] * all_data[b]
all_data[f'{a}/{b}'] = all_data[a] / all_data[b]
```

   神奇的"其他胆固醇" = =

III. Modelling

1. cross validation

```
kf = KFold(n_splits=5, shuffle=True, random_state=520)
for i, (train_index, test_index) in enumerate(kf.split(train)):
    pass
```

通常选用五折交叉验证，保证拟合程度良好、交叉验证有效。

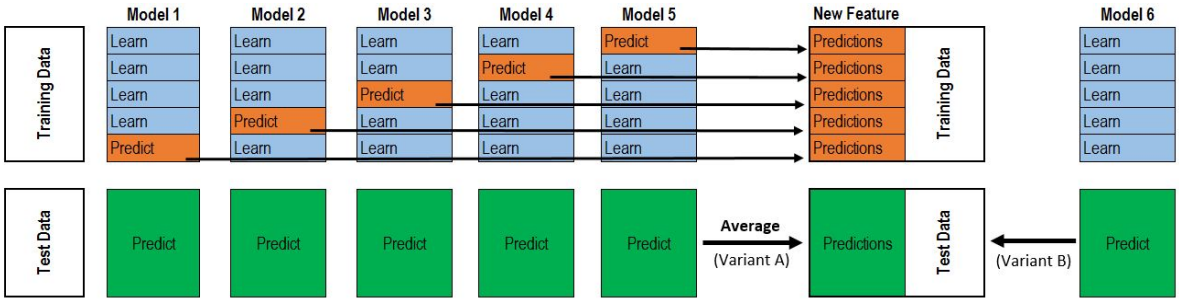
2. base models
    - LASSO Regression

      This model may be very sensitive to outliers. So we need to made it more robust on them. For that we use the sklearn's `Robustscaler()` *method on pipeline.*
    - Elastic Net Regression
    - Kernel Ridge Regression
    - Gradient Boosting Regression

      With `loss='huber'` *that makes it robust to outliers.*
    - XGBoost
    - LightGBM
    - Random Forest Regression
    - ……
3. Stacking models
    - averaging base models

```
np.column_stack([model.predict(X) for model in base_models]).mean(axis=1)
```

    - adding a Meta-model




```
stacked_averaged_models = StackingAveragedModels(
    base_models = (ENet, GBoost, KRR), meta_model = lasso)
# average Enet KRR and Gboost
# lasso as meta-model
```

    - Ensemble prediction

```
ensemble = stacked_pred * 0.70 + xgb_pred * 0.15 + lgb_pred * 0.15
```

4. evaluation function
    - mae

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

    - mse

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- rmse

$$\mathbf{RMSE}_{fo} = [\sum_{i=1}^{N}(z_{f_i} - z_{o_i})^2/N]^{1/2}$$

5. Parameter adjustment
    - Grid Search
    - Baysian hyperparameter optimization
    - ……
6. Classifier

IV. Summary

1. 评价对模型调整有很大的影响。对某个回归问题选取合适的评估函数、恰当地调整模型参数、合理的交叉验证，保证线下提升的可靠性。防止过拟合/欠拟合。
2. 线性模型的标准化、归一化、缺省值处理。
3. 特征的处理比模型优化的提升更重要。而在这方面我们没有太多进展。