

**Министерство образования и науки Российской Федерации**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего профессионального образования**  
**«Московский государственный технический университет имени Н.Э. Баумана»**  
**(МГТУ им. Н.Э. Баумана)**

<b>ФАКУЛЬТЕТ</b>	<b>«Информатики и систем управления»</b>
<b>КАФЕДРА</b>	Системы обработки информации и управления

Дисциплина «Технологии машинного обучения»

**РУБЕЖНЫЙ КОНТРОЛЬ №2**  
**«Методы построения моделей машинного обучения»**  
Вариант 21

Студент	Сахарова Е. К. ИУ5-62Б
Преподаватель	Гапанюк Ю. Е.

## 1. Задание

Группа	Метод №1	Метод №2
ИУ5-62Б	Метод опорных векторов	Случайный лес

*Задание.*

Для заданного набора данных постройте модели классификации или регрессии. Для построения моделей используйте методы 1 и 2. Оцените качество моделей на основе подходящих метрик качества. Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

*Наборы данных:*

<https://www.kaggle.com/kmldas/loan-default-prediction>

## 2. Выполнение задания

Посмотрим данные. Датасет содержит индекс (который можно не использовать в дальнейшем, т.к. он уникальный и повторяет встроенные индексы в пандасе); два бинарных значения – устроен ли человек на работу и таргет, обозначающий, что человек не платит; два числовых значения – банковский счет и ежегодный доход.

```
[ ] data = pd.read_csv('Default_Fin.csv')
data.head()
```

	Index	Employed	Bank Balance	Annual Salary	Defaulted?
0	1	1	8754.36	532339.56	0
1	2	0	9806.16	145273.56	0
2	3	1	12882.60	381205.68	0
3	4	1	6351.00	428453.88	0
4	5	1	9427.92	461562.00	0

```
[ ] data.shape

(10000, 5)
```

```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Index           10000 non-null  int64
1   Employed        10000 non-null  int64
2   Bank Balance    10000 non-null  float64
3   Annual Salary   10000 non-null  float64
4   Defaulted?      10000 non-null  int64
dtypes: float64(2), int64(3)
memory usage: 390.8 KB
```

Проверим, содержат ли колонки нулевые значения:

```
[ ] data.isnull().sum()

Index           0
Employed        0
Bank Balance    0
Annual Salary   0
Defaulted?      0
dtype: int64
```

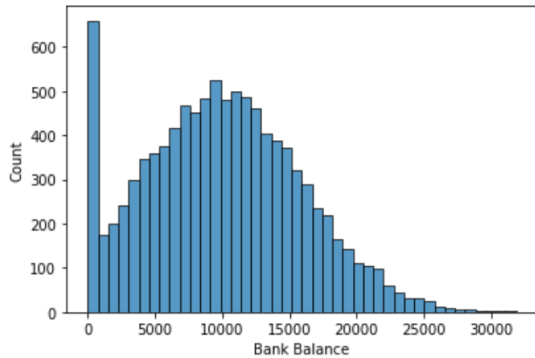
Данные полностью заполнены. Теперь нужно нормализовать их:

```
[ ] from sklearn.preprocessing import StandardScaler
```

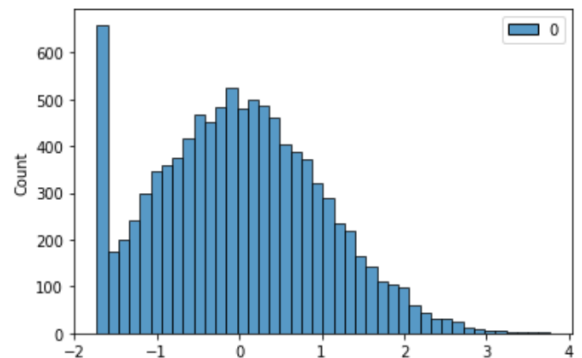
```
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)
```

```
[ ] X_scaled_col_1 = scaler.fit_transform(X[['Bank Balance']])  
X_scaled_col_2 = scaler.fit_transform(X[['Annual Salary']])
```

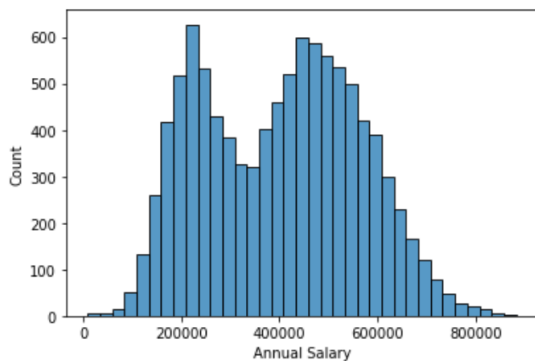
```
[ ] sns.histplot(X['Bank Balance']);
```



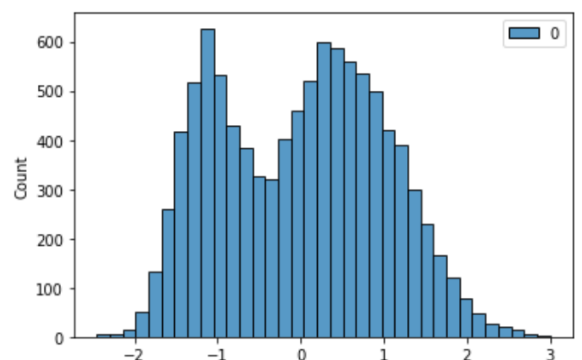
```
[ ] sns.histplot(X_scaled_col_1);
```



```
[ ] sns.histplot(X['Annual Salary']);
```



```
[ ] sns.histplot(X_scaled_col_2);
```



StandardScaler масштабирует данные таким образом, что среднее значение равно 0, а стандартное отклонение – 1.

Теперь разделим датасет на тренировочную и тестовую выборку в отношении 80% к 20%.

```
[ ] X = data[feature_cols]  
y = data['Defaulted?']
```

```
[ ] from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2, random_state=42)
```

Задача – определить, по имеющимся данным, выполнит ли человек свою часть договора по займу.

## Метод опорных векторов:

```
[ ] from sklearn.svm import SVC
```

```
svc = SVC()  
svc.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

```
[ ] y_pred_svc = svc.predict(X_test)
```

```
[ ] print('Accuracy Score: {}\nF1 Score: {}'.format(  
        accuracy_score(y_test, y_pred_svc),  
        f1_score(y_test, y_pred_svc)))
```

```
Accuracy Score: 0.968  
F1 Score: 0.2558139534883721
```

## Случайный лес:

```
[ ] from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(n_estimators=4)  
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,  
                        criterion='gini', max_depth=None, max_features='auto',  
                        max_leaf_nodes=None, max_samples=None,  
                        min_impurity_decrease=0.0, min_impurity_split=None,  
                        min_samples_leaf=1, min_samples_split=2,  
                        min_weight_fraction_leaf=0.0, n_estimators=4,  
                        n_jobs=None, oob_score=False, random_state=None,  
                        verbose=0, warm_start=False)
```

```
[ ] y_pred_rf = rf.predict(X_test)
```

```
[ ] print('Accuracy Score: {}\nF1 Score: {}'.format(  
        accuracy_score(y_test, y_pred_rf),  
        f1_score(y_test, y_pred_rf)))
```

```
Accuracy Score: 0.9645  
F1 Score: 0.297029702970297
```

Были использованы такие метрики, как Accuracy Score (подсчитывает, как часто предсказание равняется действительному значению) и F1 Score

(принимает значение от 0 до 1, поддерживает баланс между precision и recall – если точность или recall низкие, то и F1 низкая).

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

В итоге, метод опорных векторов, обучился лучше, чем модель случайного леса.