FINAL PROJECT

Amina Igibayeva, Mirolim Saidakhmatov

Outline

Dataset

Importing the dataset

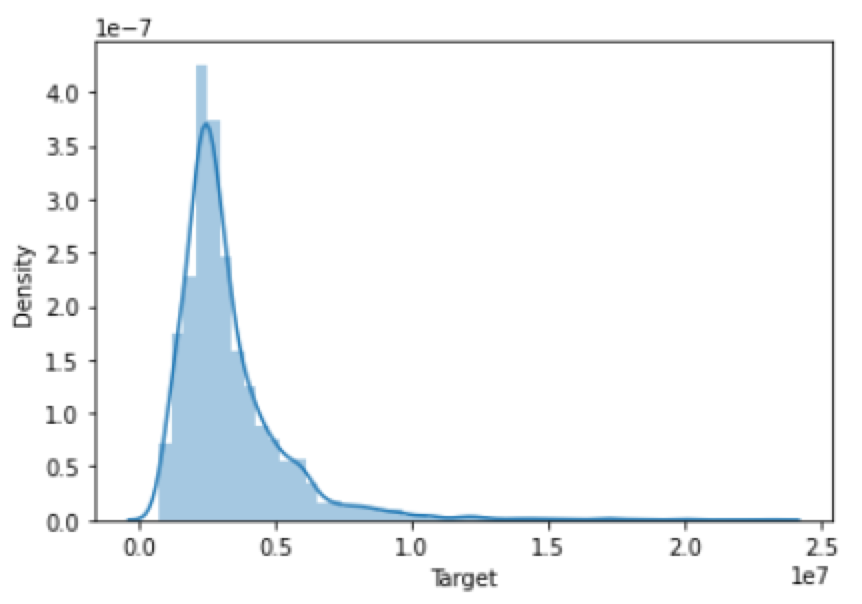
```
df = pd.read_csv('train.csv')
df
```

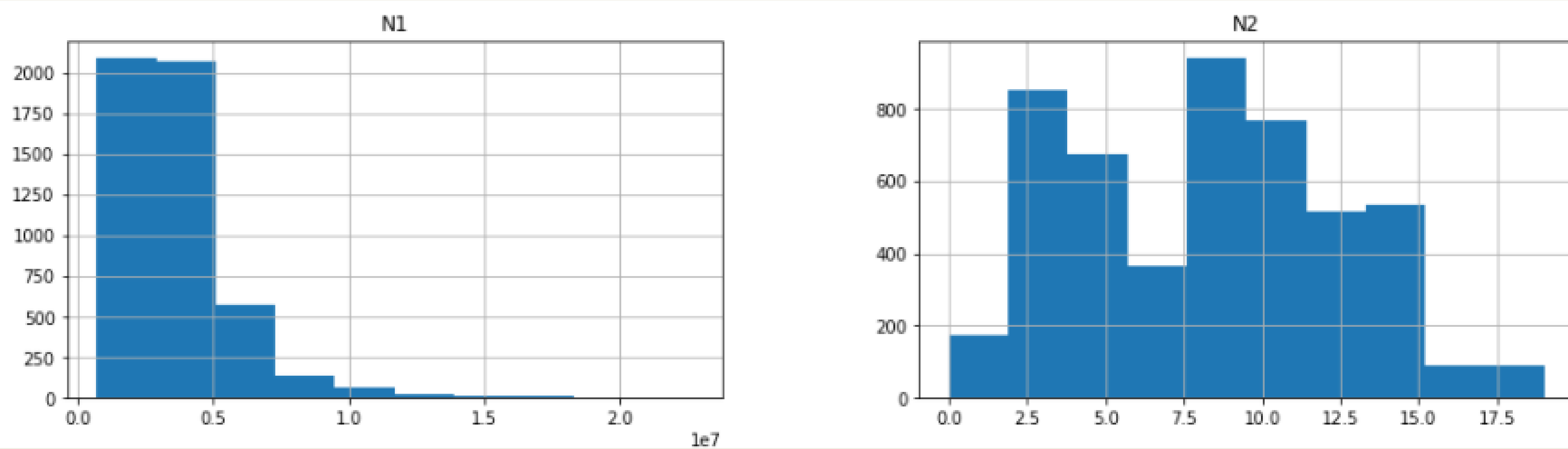
	Id	Target	N1	N2	N3	N4	N5	N6	C1	C2	C3	C4	C5	C6	C7	C8	C9
0	10000	1500000	2056164.384	11	1.8	0.0	9.0	6.0	1	M	F	V	B	S	J	T	D
1	10001	2993000	3572619.048	8	2.5	8.0	6.0	9.0	2	A	F	V	B	S	J	N	1
2	10002	9500000	9813953.488	6	3.5	2.0	9.0	0.0	1	A	B	K	B	S	W	D	D
3	10003	4056000	4529545.455	5	2.5	4.0	6.0	4.0	1	A	F	K	BG	S	4	T	1
4	10004	3543000	3823255.814	10	3.5	1.0	5.0	4.0	1	A	F	K	BG	S	4	T	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	14995	2023000	2200000.000	14	2.2	5.0	5.0	5.0	1	M	F	V	D	M	W	D	F
4996	14996	2000000	2265060.241	14	3.0	8.0	2.0	7.0	1	A	F	K	B	S	4	T	1
4997	14997	4040000	4691666.667	11	3.3	3.0	2.0	5.0	2	A	F	K	B	C	J	T	E
4998	14998	1400000	1519047.619	9	1.6	2.0	0.0	7.0	2	M	F	V	BG	S	K	L	1
4999	14999	3734000	4419753.086	10	3.5	3.0	3.0	1.0	1	A	A	K	B	C	J	N	8

5000 rows × 17 columns

EDA

Distplot of the target

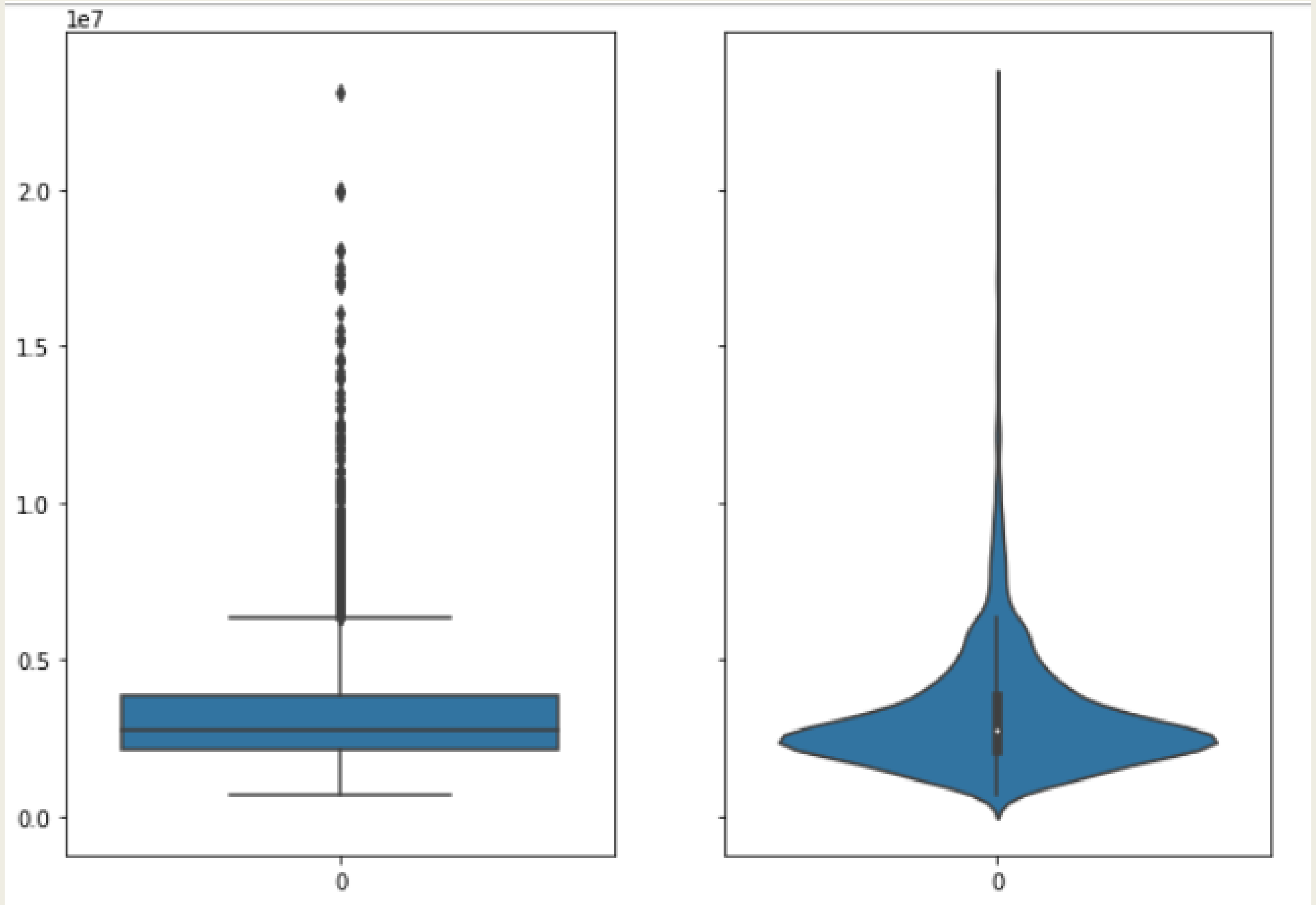




Histogram of numerical features

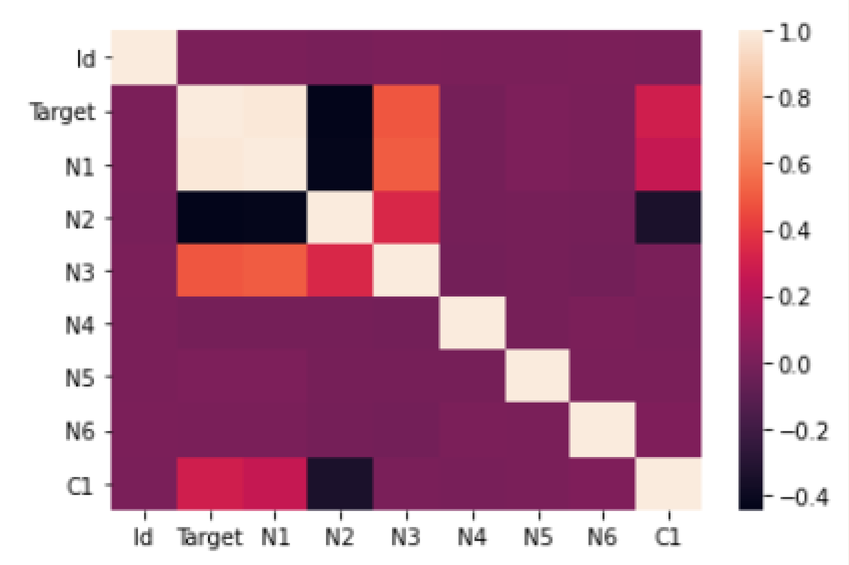
EDA

Boxplot and violinplot of the ‘Target’ column

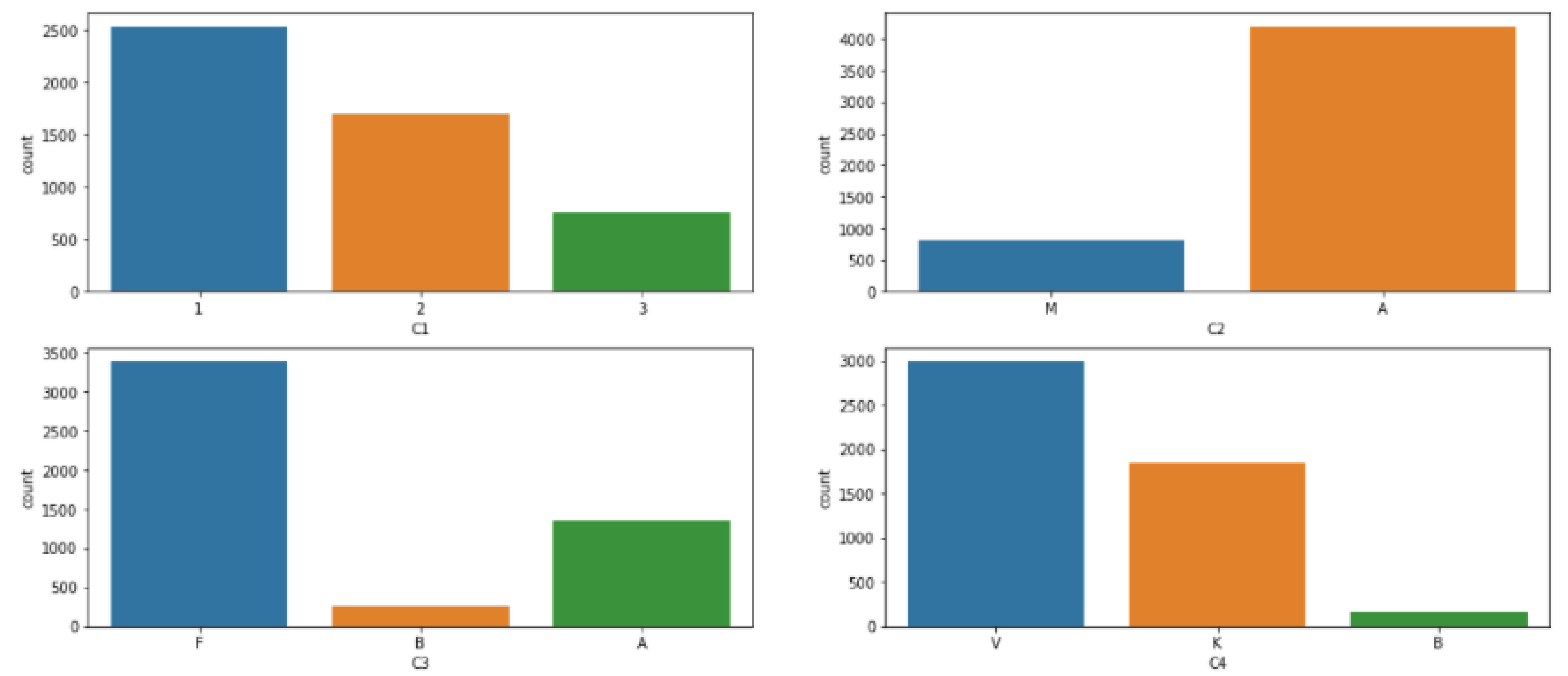


EDA

Showing countplot on categorical features



Heatmap of the correlation



Preprocessing

```
cat.remove('C1')
encoder=ce.OneHotEncoder(cols=cat, return_df=True, use_cat_names=True)
df_e = encoder.fit_transform(df_i)
```

Encoding cat. features

Filling nans

```
df_copy = df.copy()

df_i = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
df_i.isnull().sum()
```

Finding correlation, and dropping highly correlated columns, above 95%

```
# Create correlation matrix
corr_matrix = df_e.drop(columns='Target').corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

print('The highly correlated columns(above 95% of correlation) will be removed:')
print(to_drop)
# Drop features
df_e.drop(to_drop, axis=1, inplace=True)
```

Predicting using regression Models: Linear Regression, Ridge and Lasso

```
# Principle components regression
steps = [
    ('scale', MinMaxScaler()),
    ('pca', PCA(0.9449)),
    ('estimator', LinearRegression())
]
pipe = Pipeline(steps)

del X_train['Id']
pipe.fit(X_train, y_train)

y_pred_train = pipe.predict(X_train)
print(metrics.mean_squared_error(y_train, y_pred_train))
```

Train and test sets

```
train = df_e[:5000]
test = df_e[5000:]
X_train = train.drop(columns='Target')
y_train = train['Target']
X_test = test
del X_train['Id']
del X_test['Target']
del X_test['Id']

sc = [StandardScaler(), MinMaxScaler()]
md = [LinearRegression(), Ridge(), Lasso()]
ml = []
sl = []
msel = []
pcal = []
```

Running each model and storing them

```
for m in md:
    m.fit(X_train, y_train)
    y_pred_train = m.predict(X_train)
    ml.append(str(m))
    sl.append('none')
    msel.append(metrics.mean_squared_error(y_train, y_pred_train))
    pcal.append('no')
    for s in sc:
        steps = [
            ('scale', s),
            ('pca', PCA()),
            ('estimator', m)
        ]
        pipe = Pipeline(steps)
        pipe.fit(X_train, y_train)
        y_pred_train = pipe.predict(X_train)

        ml.append(str(m))
        sl.append(str(s))
        msel.append(metrics.mean_squared_error(y_train, y_pred_train))
        pcal.append('yes')
```

Predicting for test set and exporting to .csv

```
y_pred_test = lr.predict(X_test)
#metrics.mean_squared_error()
dict = {'Id': idl, 'Target': y_pred_test}
results = pd.DataFrame(dict, index=None)
results
results.to_csv('predicted.csv', index=False)
```

Evaluation metrics

	model	scaler	mse	pca
0	LinearRegression()	none	8.532147e+10	no
1	LinearRegression()	StandardScaler()	8.532229e+10	yes
2	LinearRegression()	MinMaxScaler()	8.532156e+10	yes
3	Ridge()	none	8.538519e+10	no
4	Ridge()	StandardScaler()	8.532227e+10	yes
5	Ridge()	MinMaxScaler()	9.250057e+10	yes
6	Lasso()	none	8.532158e+10	no
7	Lasso()	StandardScaler()	8.532147e+10	yes
8	Lasso()	MinMaxScaler()	8.532153e+10	yes
9	LinearRegression()	StandardScaler()	8.532147e+10	no
10	Ridge()	StandardScaler()	8.532227e+10	no
11	Lasso()	StandardScaler()	8.532147e+10	no
12	LinearRegression()	MinMaxScaler()	8.532161e+10	no
13	Ridge()	MinMaxScaler()	9.250057e+10	no
14	Lasso()	MinMaxScaler()	8.532158e+10	no

Conclusion

- We have seen cleaning, shaping, EDA, preprocessing and PCA on our data. Then we trained and tested our data on different models with different scalings:
- -Linear Regression
- -Ridge
- -Lasso
- And we have the best performance with Linear Regression.