In [2]:
```
# Пратика 1
import pandas as pd
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction import DictVectorizer
from sklearn.linear_model import Ridge
from scipy.sparse import hstack, csr_matrix

def preprocess_data(df):
    df_processed = df.copy()
    df_processed['FullDescription'] = df_processed['FullDescription'].str.lower(
    df_processed['FullDescription'] = df_processed['FullDescription'].replace('[
    df_processed = df_processed.fillna({'LocationNormalized': 'nan', 'ContractTi
    return df_processed

train = pd.read_csv('salary-train.csv')
train_processed = preprocess_data(train)
tfidf = TfidfVectorizer(min_df=5)
X_text = tfidf.fit_transform(train_processed['FullDescription'])

categorical_data = []
for i in range(len(train_processed)):
    row_dict = {
        'LocationNormalized': train_processed['LocationNormalized'].iloc[i],
        'ContractTime': train_processed['ContractTime'].iloc[i]
    }
    categorical_data.append(row_dict)
dict_vectorizer = DictVectorizer()
X_cat = dict_vectorizer.fit_transform(categorical_data)
X_combined = hstack([X_text, X_cat])

y = train_processed['SalaryNormalized']
mask = y.notna()
indices = mask.values
X_combined = X_combined[indices]
y = y[mask]
ridge = Ridge(alpha=1, random_state=241)

ridge.fit(X_combined, y)

test = pd.read_csv('salary-test-mini.csv')
test_processed = preprocess_data(test)
X_test_text = tfidf.transform(test_processed['FullDescription'])
test_categorical_data = []
for i in range(len(test_processed)):
    row_dict = {
        'LocationNormalized': test_processed['LocationNormalized'].iloc[i],
        'ContractTime': test_processed['ContractTime'].iloc[i]
    }
    test_categorical_data.append(row_dict)
X_test_cat = dict_vectorizer.transform(test_categorical_data)
X_test_combined = hstack([X_test_text, X_test_cat])

predictions = ridge.predict(X_test_combined)

answer = " ".join([str(round(pred, 2)) for pred in predictions])
print(answer)
```

```
56573.82 37196.34
```

```
In [ ]:
```

In [8]:
```python
# Практика 2
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from scipy.stats import pearsonr

prices = pd.read_csv('close_prices.csv')
pca_10 = PCA(n_components=10)

X = prices.iloc[:, 1:]
pca_10.fit(X)
explained_variance_ratio = pca_10.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
n_components_90 = np.argmax(cumulative_variance_ratio >= 0.9) + 1

first_component = X_pca[:, 0]

djia = pd.read_csv('djia_index.csv')

djia_column_name = "^DJI"
correlation, p_value = pearsonr(first_component, djia[djia_column_name])

first_component_weights = pca_10.components_[0]
company_names = X.columns
weights_df = pd.DataFrame({
    'Company': company_names,
    'Weight': first_component_weights
})
max_weight_idx = np.argmax(np.abs(first_component_weights))
company_with_max_weight = company_names[max_weight_idx]
max_weight = first_component_weights[max_weight_idx]


print(f"1. Количество компонент для объяснения 90% дисперсии: {n_components_90}"
print(f"2. Корреляция Пирсона между первой компонентой и индексом Доу-Джонса: {c
print(f"3. Компания с наибольшим весом в первой компоненте: {company_with_max_we
```

```
1. Количество компонент для объяснения 90% дисперсии: 4
2. Корреляция Пирсона между первой компонентой и индексом Доу-Джонса: 0.91
3. Компания с наибольшим весом в первой компоненте: V
```

In [ ]: