```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score



filename = '/content/drive/MyDrive/Admission_Predict.csv'
df = pd.read_csv(filename)


df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
info = df.info()
print(info)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 2   TOEFL Score        400 non-null    int64  
 3   University Rating  400 non-null    int64  
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64  
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.3 KB
None
```

```
y = df.iloc[:, 8]  # 8 — индекс столбца "Chance of Admit"

X = df.iloc[:, 1:8]  # от 1 до 7 включительно (Serial No. исключаем)


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


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


svr_model = SVR(kernel='rbf')
svr_model.fit(X_train_scaled, y_train)
y_pred_svr = svr_model.predict(X_test_scaled)


gb_model = GradientBoostingRegressor(n_estimators=200, learning_rate=0.1, max_depth=3, random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)


print("SVR - MSE:", mean_squared_error(y_test, y_pred_svr))
print("SVR - R²:", r2_score(y_test, y_pred_svr))

print("\nGB - MSE:", mean_squared_error(y_test, y_pred_gb))
print("GB - R²:", r2_score(y_test, y_pred_gb))
```

```
SVR - MSE: 0.006203249044866006
SVR - R²: 0.7597814848647667
```

```
GB - MSE: 0.005861843998636167
GB - R²: 0.7730022684689464
```

Выводы

- Градиентный бустинг показал лучшие результаты по сравнению с SVM.
- Обе модели показали высокую точность, что говорит о том, что выбранные признаки действительно содержат полезную информацию для прогнозирования шансов на поступление
- SVM чувствителен к масштабу данных, поэтому нормализация важна.
- Градиентный бустинг работает без нормализации, но может быть склонен к переобучению при увеличении глубины деревьев.