دانشکده فنی و مهندسی
گروه مهندسی کامپیوتر

پروژه داده کاوی

ملیکا قنبری

شماره دانشجویی

9914162123

استاد درس

دکتر فاطمه باقری

نیمسال اول سال تحصیلی ۱۴۰۳-۱۴۰۴

۱- مجموعه داده

این دیتاست به پیش‌بینی احتمال بروز سکته مغزی بر اساس ویژگی‌های دموگرافیکی و پزشکی افراد می‌پردازد. اطلاعات این دیتاست شامل ویژگی‌های مختلفی است که عوامل مؤثر در بروز سکته را توصیف می‌کنند.

نوع متغیر هدف: متغیر دودویی (Binary Classification).

- موضوع داده Stroke Prediction
- تعداد کل ۵۱۱۰ نمونه
- تعداد ویژگی : ۱۱ ویژگی و ۱ متغیر هدف
- ویژگی ها شامل id ، gender ، age ، hypertension ، ever_married ، work_type ، Residence_type ، avg_glucose_level ، bmi ، smoking_status و stroke

۱-۱- مشخصات مجموعه داده

جدول توصیف مجموعه داده

ردیف	ویژگی	توصیف
۱	id	شناسه منحصربه‌فرد هر فرد (برای تحلیل استفاده نمی‌شود)
۲	gender	جنسیت فرد (مقادیر ممکن: Male, Female, Other)
۳	age	سن فرد بر حسب سال
۴	hypertension	آیا فرد دارای فشار خون بالا است؟ (۰ = خیر، ۱ = بله).
۵	heart_disease	آیا فرد به بیماری قلبی مبتلا است؟ (۰ = خیر، ۱ = بله).
۶	ever_married	آیا فرد تاکنون ازدواج کرده است؟ (Yes, No)
۷	work_type	وع شغل فرد (مقادیر ممکن: children, Govt_job, Never_worked, Private, Self-employed).
۸	Residence_type	نوع محل سکونت فرد (Urban = شهری، Rural = روستایی).
۹	avg_glucose_level	میانگین سطح گلوکز خون فرد (mg/dL).
۱۰	bmi	شاخص توده بدنی فرد (Body Mass Index).
۱۱	smoking_status	وضعیت استعمال دخانیات فرد (formerly smoked, never smoked, smokes, Unknown).

۱۲	stroke	متغیر هدف، نشان‌دهنده بروز سکته (۱ = سکته، ۰ = بدون سکته).

۱-۲- نمونه هایی از مجموعه داده

id	gender	age	hypertension	heart_disease	...

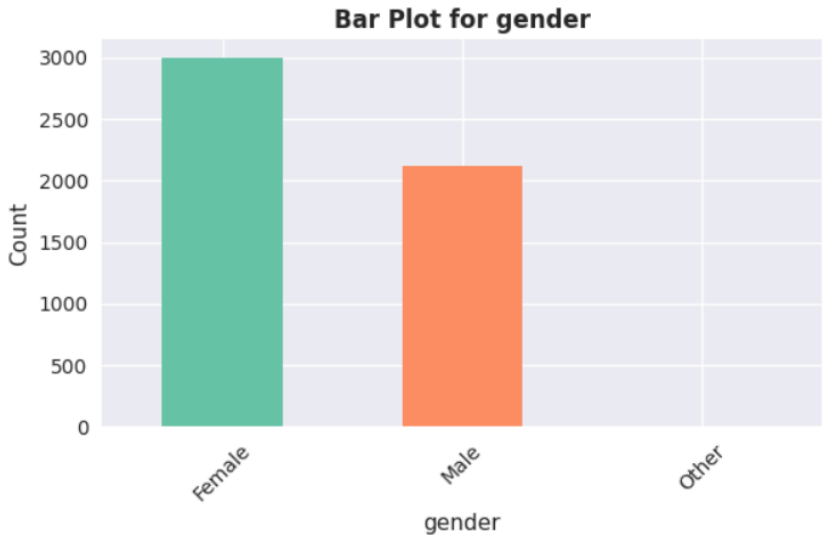
۱-۳- مشخصات اولیه ویژگی های مجموعه داده

ویژگی	تعداد	ماکزیمم مقدار	مینیمم مقدار	مقدار میانه	تعداد داده های ناموجود
avg_glucose_level	۵۱۱۰	۲۷۱.۷۴	۵۵.۱۲	۹۱.۸۸۴۹۹۹۹۹	۰
BMI	۵۱۱۰	۹۷.۶	۱۰.۳	۲۸.۱	۲۰۱
age	۵۱۱۰	۸۲	۰.۰۸	۴۵	۰
جنسیت	۵۱۱۰	۲۹۹۴ مورد → زن ۲۱۱۵ مورد → مرد			۱

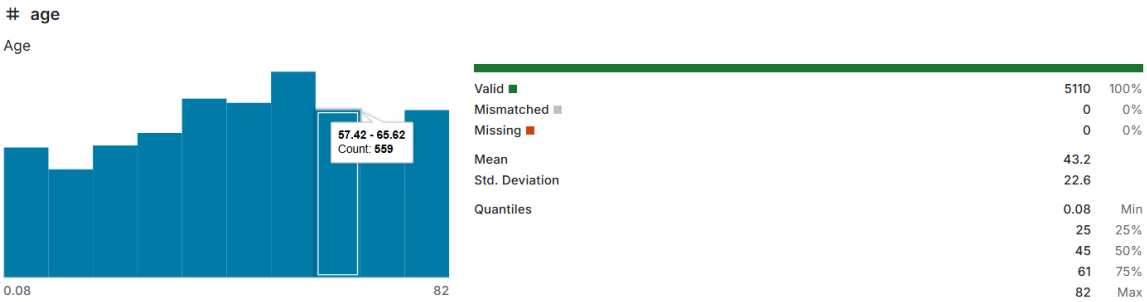
۱-۴- نمودارهای توصیف داده

در این بخش نمودارهایی که به توصیف داده ها کمک می کنند قرار می گیرند. به عنوان مثال برای هر کدام از ویژگی های اسمی نموداری طراحی شود که تعداد نمونه های هر کدام از مقادیر اسمی را نشان دهد و در عین حال قابل مقایسه با تعداد مقادیر دیگر باشد.

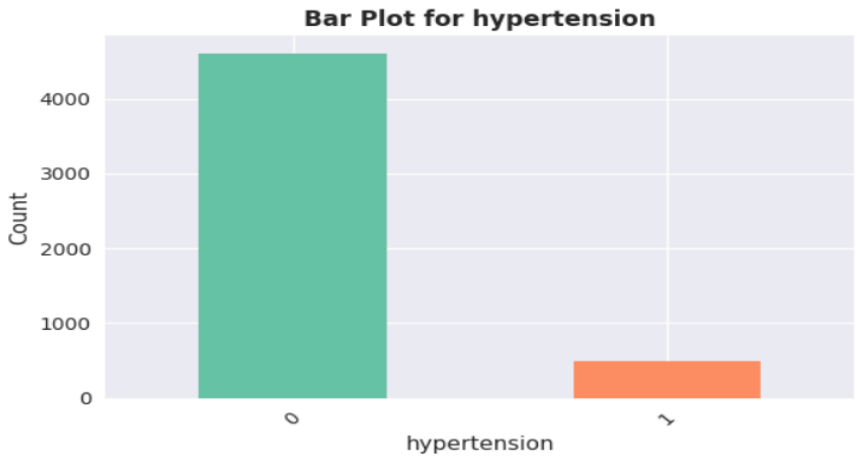
به عنوان مثال برای جنسیت در مجموعه داده بالا می توان نمودار زیر را رسم کرد.



شکل ۱: فراوانی ویژگی جنسیت در مجموعه داده

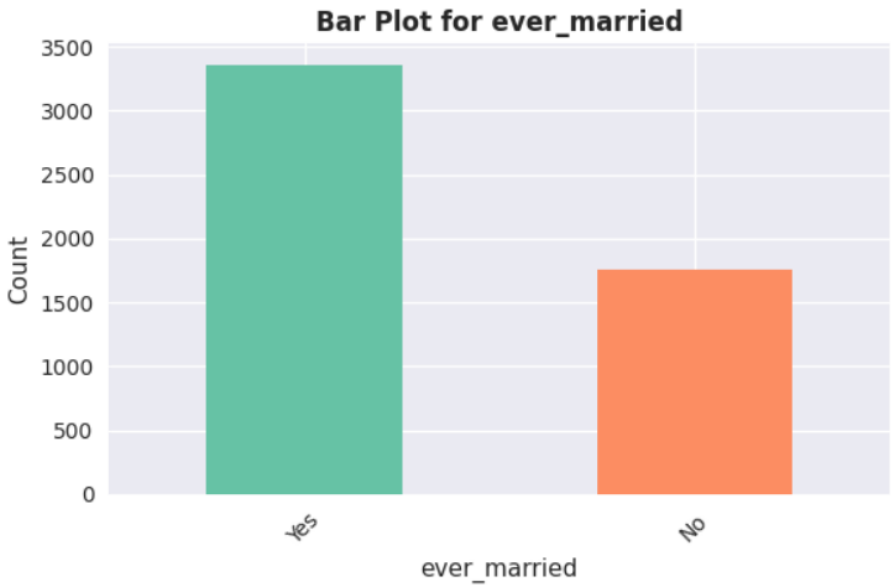


شکل ۲ : فراوانی ویژگی سن در مجموعه داده

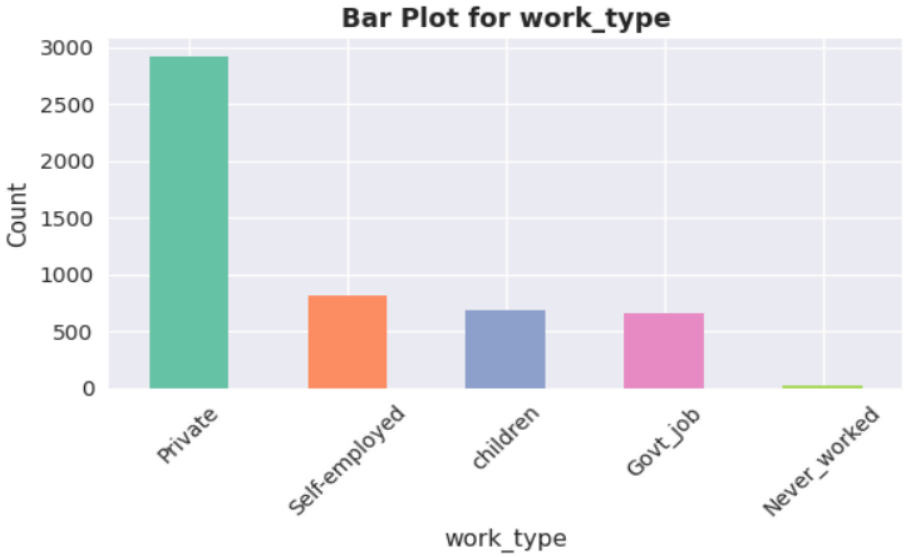


شکل ۳ : فراوانی ویژگی Hypertension در مجموعه داده

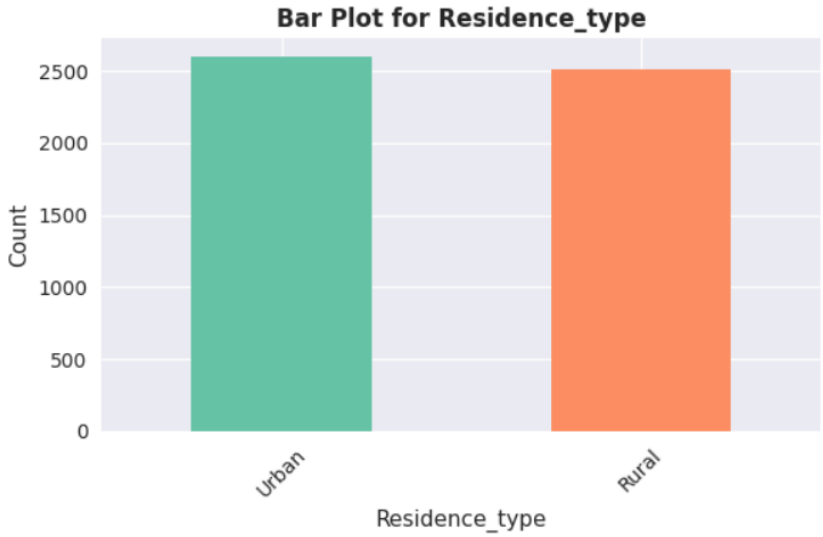
شکل ۴ : فراوانی ویژگی Heart_disease در مجموعه داده



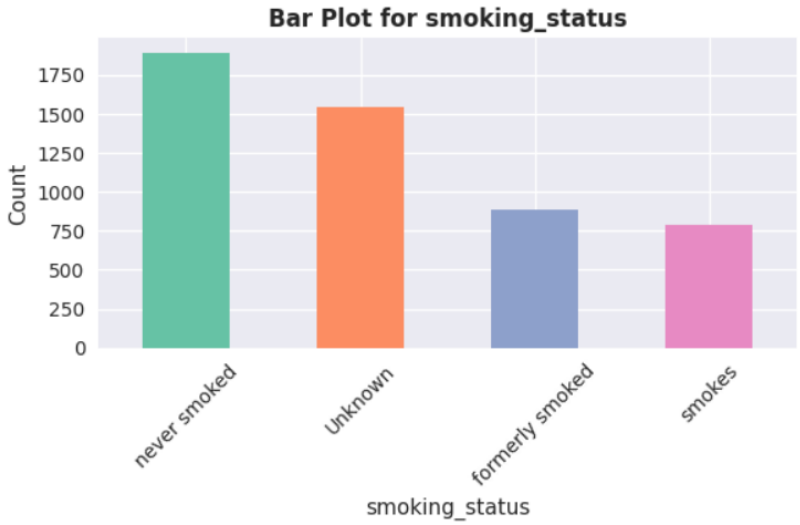
شکل ۵ : فراوانی ویژگی ever_married در مجموعه داده



شکل ۶ : فراوانی ویژگی work_type در مجموعه داده



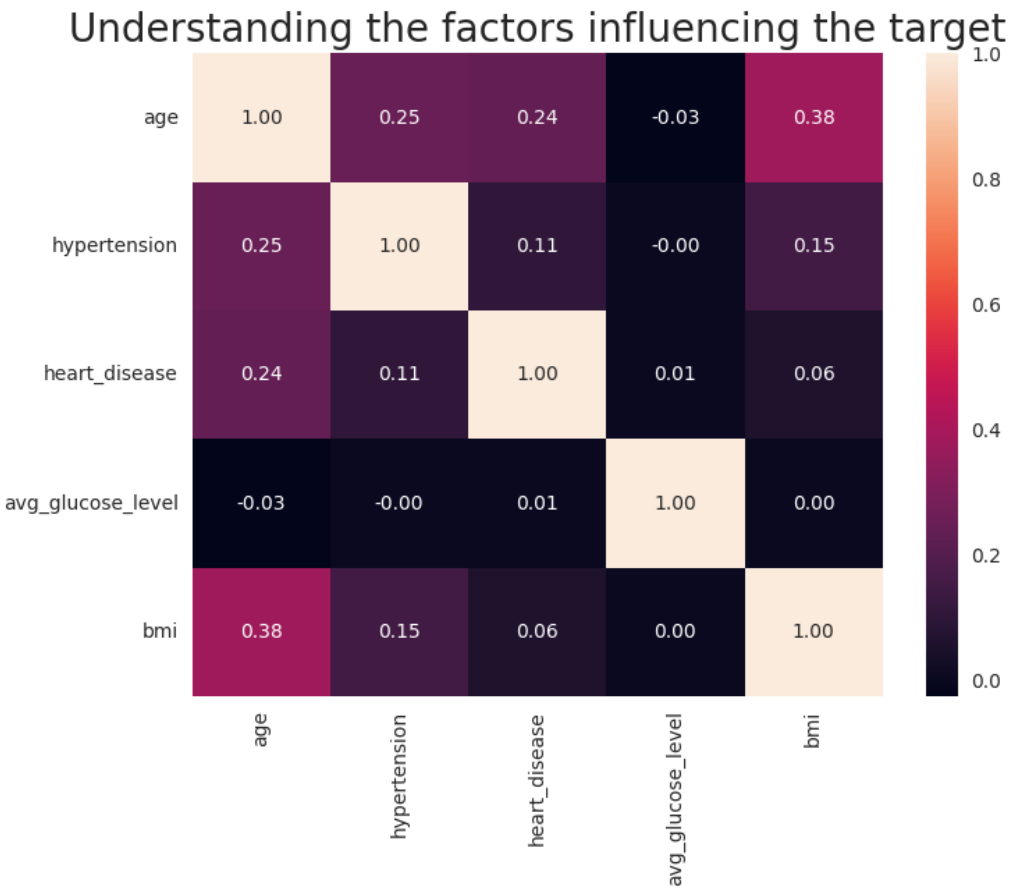
شکل ۷ : فراوانی ویژگی Residence_type در مجموعه داده



شکل ۸ : فراوانی ویژگی smoking_status در مجموعه داده

شکل ۹ : فراوانی ویژگی stroke در مجموعه داده

و در پایان اشکال، نمودار وابستگی ویژگی های مجموعه داده (heatmap) را قرار دادیم . این نمودار یکی از ابزارهای بصری‌سازی داده‌هاست که با استفاده از ماتریس همبستگی بین ویژگی‌های مجموعه داده، میزان ارتباط (وابستگی) آن‌ها را نمایش می‌دهد.به عوان مثال شکل زیر این نمودار را برای مجموعه داده Stroke Prediction نشان می دهد.



شکل ۱۰ : نقشه حرارتی ضرایب همبستگی پیرسون برای ۵ ویژگی

۵ – ۱- پیش پردازش

ابتدا داده ها را از منبع مربوطه بارگذاری کردیم و با استفاده از کتابخانه pandas به قالب DataFrame تبدیل کردیم . یکی از مراحل مهم در پیش‌پردازش، بررسی داده‌ها برای وجود مقادیر گم‌شده (Null) است که در داده ما مقادیر گم شده وجود نداشت . سپس، به منظور آماده‌سازی داده‌ها برای آموزش مدل، ویژگی‌های ورودی و خروجی را مشخص کردیم و داده‌ها به دو دسته آموزشی و تست تقسیم شدند تا مدل بتواند با داده‌های آموزشی یاد بگیرد و با داده‌های تست ارزیابی شود

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('/content/gdrive/MyDrive/healthcare-dataset-stroke-data.csv')
# Drop the 'id' column as it is not relevant for analysis
data.drop(columns=['id'], inplace=True)
```

کتابخانه‌های اصلی:

Pandas :

برای کار با داده‌های جدولی.(DataFrames)

عملیات‌هایی مانند بارگذاری داده، مدیریت مقادیر گمشده، و حذف ستون‌ها.

Numpy :

برای انجام عملیات ریاضی و محاسباتی.

در این کد برای مدیریت مقادیر غیرمنفی و تبدیل داده‌ها استفاده شده است.

کتابخانه‌های یادگیری ماشین sklearn

sklearn.model_selection

train_test_split برای تقسیم داده‌ها به مجموعه‌های آموزشی و آزمایشی.

Sklearn.preprocessing

LabelEncoder برای تبدیل ویژگی‌های دسته‌ای به مقادیر عددی.

StandardScalerبرای استانداردسازی داده‌ها به مقیاس یکسان.

Sklearn.naive_bayes

GaussianNB برای اجرای مدل Naïve Bayes با فرض داده‌های پیوسته و توزیع گاوسی.

MultinomialNBبرای داده‌های گسسته یا غیرمنفی.

BernoulliNB برای داده‌های باینری.

Sklearn.svm

SVC برای پیاده‌سازی ماشین بردار پشتیبان (SVM) با کرنل‌های مختلف.

sklearn.metrics

classification_report برای ایجاد گزارش دقت، فراخوانی، و F1-Score.

confusion_matrix برای محاسبه و نمایش ماتریس سردرگمی.

برای راحتی کار داده را در google drive بارگذاری کردیم و ستون id را حذف کردیم چون مربوط با کار ما نیست .

۶–۱- مقادیر گم شده

```
data['bmi'].fillna(data['bmi'].median(), inplace=True)
```

در اینجا ما مقادیر گم شده در bmi را با میانه اش پر می میکنیم

۷–۱- تحلیل

```
label_encoders = {}
categorical_columns = ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le
```

حالا با استفاده از LabelEncoder ستون هایی که اسمی هستند را رمزگذاری می کنیم یعنی ویژگی‌های اسمی مثل جنسیت یا نوع کار، با استفاده از LabelEncoder به مقادیر عددی تبدیل شده‌اند تا برای الگوریتم‌ها قابل استفاده باشند.

۸–۱- تفکیک ویژگی‌ها و متغیر هدف

```
X = data.drop(columns=['stroke'])
y = data['stroke']
```

X : شامل ویژگی‌های مستقل (پیش‌بینی‌کننده) است.

Y : متغیر هدف (آیا فرد دچار سکته شده یا نه) است.

۹–۱– استاندارد سازی داده ها

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

داده‌های عددی مقیاس‌بندی شده‌اند تا تمام ویژگی‌ها مقیاس یکسانی داشته باشند. این کار برای مدل‌هایی مانند SVM ضروری است

۱۰– ۱– تقسیم داده‌ها به مجموعه‌های آموزشی و تست

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42, stratify=y)
```

در این مرحله، داده‌ها به دو مجموعه آموزشی و تست تقسیم شدند. با استفاده از کد X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)، ۷۰٪ از داده‌ها به عنوان داده‌های آموزشی (X_train و y_train) و ۳۰٪ به عنوان داده‌های تست (X_test و y_test) انتخاب شدند. این تقسیم‌بندی به ما این امکان را می‌دهد که مدل را با داده‌های آموزشی آموزش دهیم و سپس عملکرد آن را با داده‌های تست ارزیابی کنیم. استفاده از random_state=42 باعث می‌شود تا نتایج تکراری و قابل اعتماد حاصل شود و هر بار که کد اجرا می‌شود، تقسیم‌بندی داده‌ها یکسان باشد . با stratify=y، نسبت داده‌های مثبت و منفی در دو مجموعه یکسان نگه داشته شده است.

۱۱– ۱– توابع کمکی

```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)
    matrix = confusion_matrix(y_test, y_pred)
    return report, matrix
```

از این تابع برای ارزیابی مدل استفاده می کنیم . پیش‌بینی‌ها انجام شده و معیارهای ارزیابی مثل دقت(Precision) ، فراخوانی (Recall)و F1-Score محاسبه می‌شوند. ماتریس سردرگمی (Confusion Matrix) نیز برای بررسی عملکرد مدل ایجاد می‌شود.

۱۲-۱- پیاده‌سازی Naive Bayes

```
# Initialize and train Naive Bayes models
results_nb = {}
models_nb = {
    "GaussianNB": GaussianNB(),
    "MultinomialNB": MultinomialNB(),
    "BernoulliNB": BernoulliNB()
}

# Train and evaluate GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
report_gnb, matrix_gnb = evaluate_model(gnb, X_test, y_test)
results_nb["GaussianNB"] = {"classification_report": report_gnb, "confusion_matrix": matrix_gnb}
```

Naive Bayes یک الگوریتم یادگیری ماشین مبتنی بر آمار است که برای مسائل طبقه‌بندی استفاده می‌شود. این الگوریتم بر اساس نظریه بیز عمل می‌کند و فرض می‌کند که ویژگی‌ها به‌طور مستقل از یکدیگر برای هر کلاس توزیع می‌شوند. این مدل به‌ویژه در مسائل طبقه‌بندی با داده‌های گسسته و بزرگ، مانند تحلیل متن (تشخیص اسپم یا دسته‌بندی اسناد) عملکرد بسیار خوبی دارد. Naive Bayes معمولاً با سه نوع مختلف از توزیع‌ها پیاده‌سازی می‌شود:(GaussianNB برای داده‌های پیوسته)، MultinomialNB (برای داده‌های گسسته) و(BernoulliNB)برای داده‌های باینری . در کد اجرا شده، ابتدا داده‌های متنی کدگذاری شدند تا به صورت عددی تبدیل شوند و سپس داده‌ها به دو بخش آموزش و تست تقسیم شدند. در مرحله بعد، ویژگی‌ها با استفاده از StandardScaler نرمال شدند تا مقیاس‌های متفاوت ویژگی‌ها تاثیر منفی بر روی مدل نگذارند. سپس مدل GaussianNB از کتابخانه sklearn برای یادگیری بر روی داده‌های آموزش فیت شد. در نهایت، مدل آموزش دیده برای پیش‌بینی برچسب‌ها (کلاس‌ها) بر روی داده‌های تست استفاده شد و ارزیابی‌های لازم با استفاده از معیارهای دقت، گزارش طبقه‌بندی و ماتریس سردرگمی انجام شد تا عملکرد مدل بررسی شود. مدل با داده‌های استاندارد شده آموزش می‌بیند

```
# Prepare non-negative data for MultinomialNB and BernoulliNB
X_non_negative = scaler.inverse_transform(X_train)  # Inverse transform to original scale
X_test_non_negative = scaler.inverse_transform(X_test)
X_non_negative = np.maximum(0, X_non_negative)
X_test_non_negative = np.maximum(0, X_test_non_negative)
```

```
# Train and evaluate MultinomialNB and BernoulliNB
for name in ["MultinomialNB", "BernoulliNB"]:
    model = models_nb[name]
    model.fit(X_non_negative, y_train)
    report, matrix = evaluate_model(model, X_test_non_negative, y_test)
    results_nb[name] = {"classification_report": report, "confusion_matrix": matrix}
```

حالا باید داده‌هارا برای MultinomialNB و BernoulliNB پیاده سازی کنیم . MultinomialNB و BernoulliNB فقط مقادیر غیرمنفی را می‌پذیرند . بنابراین داده‌ها به مقیاس اصلی بازگردانده شده و مقادیر منفی صفر شده‌اند و در اخر مدل را آموزش می دهیم

۱۳- ۱- پیاده‌سازی SVM

```
results_svm = {}
kernels = ['linear', 'poly', 'rbf']

for kernel in kernels:
    svm_model = SVC(kernel=kernel, probability=True, random_state=42)
    svm_model.fit(X_train, y_train)
    report, matrix = evaluate_model(svm_model, X_test, y_test)
    results_svm[kernel] = {"classification_report": report, "confusion_matrix": matrix}
```

Support Vector Machine (SVM) یک الگوریتم یادگیری نظارت‌شده است که برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. این الگوریتم سعی می‌کند بهترین خط یا صفحه (در فضای ویژگی‌های با ابعاد بالا) را پیدا کند که داده‌ها را به دو کلاس مختلف تقسیم کند. در واقع، SVM به دنبال پیدا کردن هایپرهوا (Hyperplane) است که بیشترین فاصله را از نزدیک‌ترین داده‌ها به هر کلاس داشته باشد. این نقاط نزدیک به هایپرهوا که در حاشیه مرزی قرار دارند، پشتیبان‌ها (Support Vectors) نامیده می‌شوند. SVM می‌تواند با استفاده از هسته‌ها (Kernels) برای حل مسائل غیرخطی کاربردی باشد، به این معنی که از تبدیل‌های غیرخطی برای مشخص کردن مرزهای تصمیم در داده‌های پیچیده‌تر استفاده می‌کند. این الگوریتم معمولاً در مسائل با داده‌های بزرگ و پیچیده، که به دقت بالایی نیاز دارند، بسیار موثر است. سه نوع کرنل (تابع نگاشت) برای SVM استفاده شده است:

linear برای داده‌های خطی

poly برای داده‌های چندجمله‌ای

Rbf برای داده‌های غیرخطی

و در آخر مدل SVM با هر کرنل آموزش دیده و ارزیابی می‌شود

۱۴- ۱- پیاده‌سازی KNN

```
k_values = range(1, 21)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies.append(accuracy_score(y_test, y_pred))
```

KNN یکی از ساده‌ترین و محبوب‌ترین الگوریتم‌های یادگیری ماشین است که در مسائل طبقه‌بندی (Classification) و رگرسیون (Regression) استفاده می‌شود. این الگوریتم از یادگیری مبتنی بر نمونه (Instance-based Learning) استفاده می‌کند، به این معنا که در حین آموزش، مدل هیچ پارامتری یاد نمی‌گیرد؛ بلکه داده‌ها را ذخیره کرده و در هنگام پیش‌بینی، تصمیم‌گیری می‌کند.

range(1, 21) : مقادیر KKKاز ۱ تا ۲۰ بررسی می‌شوند.

KNeighborsClassifier(n_neighbors=k) : یک مدل KNN با تعداد همسایه مشخص KKK ایجاد می‌شود.

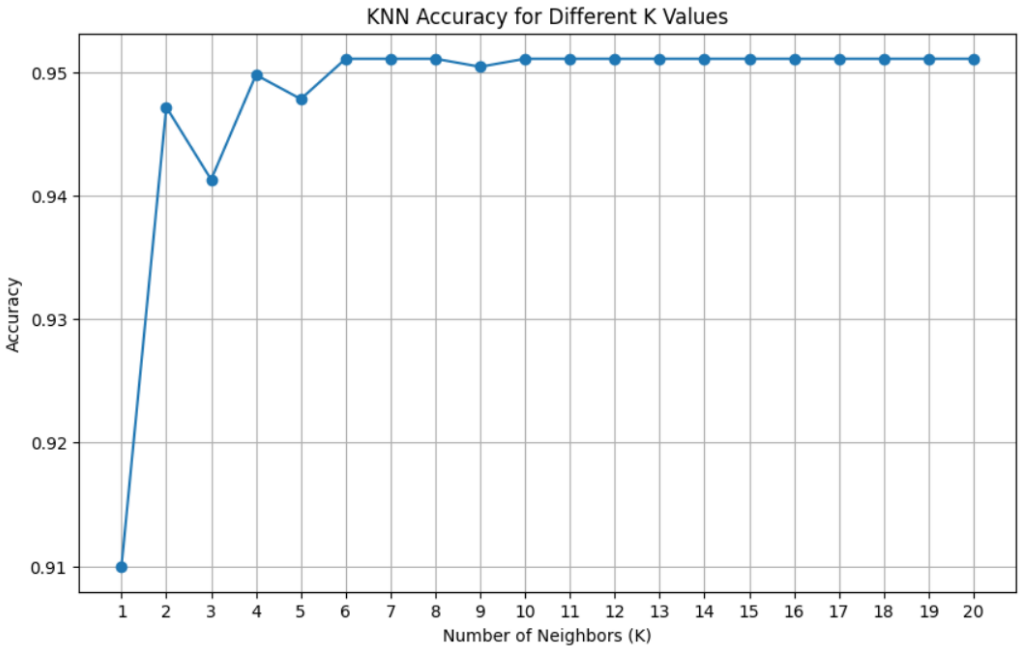
fit : مدل روی داده‌های آموزشی آموزش داده می‌شود.

predict : پیش‌بینی روی داده‌های تست انجام می‌شود.

accuracy_score : دقت مدل برای مقدار مشخص KKK محاسبه و ذخیره می‌شود.

```
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracies, marker='o')
plt.title('KNN Accuracy for Different K Values')
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Accuracy')
plt.xticks(k_values)
plt.grid()
plt.show()
```

نمودار تغییرات دقت با مقادیر مختلف KKKرسم می‌شود تا بهترین مقدار KKK انتخاب شود



شکل ۱۱ : نمودار دقت برای مقادیر مختلف K

۱۵- ۱- ارزیابی مدل ها

قسمت ارزیابی مدل‌ها شامل دو نوع خروجی است: گزارش طبقه‌بندی (Classification Report) و ماتریس سردرگمی (Confusion Matrix). این خروجی‌ها برای تحلیل عملکرد مدل‌ها در پیش‌بینی داده‌های آزمایشی استفاده می‌شوند.

گزارش طبقه‌بندی شامل معیارهای زیر است که برای هر کلاس (مثلاً ۰ و ۱ برای متغیر هدف stroke) محاسبه می‌شود

Precision (دقت) : درصد پیش‌بینی‌های درست از میان تمام نمونه‌هایی که مدل به عنوان مثبت پیش‌بینی کرده است.

Recall (فراخوانی) : درصد نمونه‌های مثبت واقعی که مدل به درستی شناسایی کرده است

F1-Score : میانگین هماهنگ دقت و فراخوانی. معیاری متعادل برای ارزیابی مدل در شرایطی که تعادل بین Precision و Recall مهم است.

Support : تعداد نمونه‌های واقعی هر کلاس در داده‌های آزمایشی

❊ مدل Naive Bayes

```
print("Naive Bayes Results:")
for model_name, result in results_nb.items():
    print(f"\nModel: {model_name}")
    print("Classification Report:")
    print(pd.DataFrame(result["classification_report"]))
    print("Confusion Matrix:")
    print(result["confusion_matrix"])
```

```
Model: GaussianNB
Classification Report:
                    0          1  accuracy    macro avg  weighted avg
precision    0.965595   0.147959  0.861057     0.556777      0.925593
recall       0.885460   0.386667  0.861057     0.636063      0.861057
f1-score     0.923792   0.214022  0.861057     0.568907      0.889068
support    1458.000000  75.000000  0.861057  1533.000000   1533.000000
Confusion Matrix:
[[1291  167]
 [  46   29]]

Model: MultinomialNB
Classification Report:
                    0          1  accuracy    macro avg  weighted avg
precision    0.965462   0.111111  0.804958     0.538286      0.923664
recall       0.824417   0.426667  0.804958     0.625542      0.804958
f1-score     0.889382   0.176309  0.804958     0.532845      0.854496
support    1458.000000  75.000000  0.804958  1533.000000   1533.000000
Confusion Matrix:
[[1202  256]
 [  43   32]]

Model: BernoulliNB
Classification Report:
                    0          1  accuracy    macro avg  weighted avg
precision    0.951475   0.125000  0.947162     0.538238      0.911041
recall       0.995199   0.013333  0.947162     0.504266      0.947162
f1-score     0.972846   0.024096  0.947162     0.498471      0.926430
support    1458.000000  75.000000  0.947162  1533.000000   1533.000000
Confusion Matrix:
[[1451    7]
 [  74    1]]
```

❄ مدل SVM

حال نوبت ارزیابی هست مقادیر بالا را به ما نمایش می دهد :

```
# Print results for SVM
print("\nSVM Results:")
for kernel, result in results_svm.items():
    print(f"\nKernel: {kernel}")
    print("Classification Report:")
    print(pd.DataFrame(result["classification_report"]))
    print("Confusion Matrix:")
    print(result["confusion_matrix"])
```

خروجی به صورت زیر است :

```
SVM Results:

Kernel: linear
Classification Report:
                    0     1  accuracy    macro avg  weighted avg
precision    0.951076   0.0  0.951076     0.475538      0.904546
recall       1.000000   0.0  0.951076     0.500000      0.951076
f1-score     0.974925   0.0  0.951076     0.487462      0.927228
support    1458.000000  75.0  0.951076  1533.000000   1533.000000
Confusion Matrix:
[[1458    0]
 [  75    0]]

Kernel: poly
Classification Report:
                    0          1  accuracy    macro avg  weighted avg
precision    0.951571   0.200000  0.949119     0.575785      0.914801
recall       0.997257   0.013333  0.949119     0.505295      0.949119
f1-score     0.973878   0.025000  0.949119     0.499439      0.927455
support    1458.000000  75.000000  0.949119  1533.000000   1533.000000
Confusion Matrix:
[[1454    4]
 [  74    1]]
```

```
Kernel: rbf
Classification Report:
                     0     1  accuracy    macro avg  weighted avg
precision     0.951076   0.0  0.951076     0.475538      0.904546
recall        1.000000   0.0  0.951076     0.500000      0.951076
f1-score      0.974925   0.0  0.951076     0.487462      0.927228
support    1458.000000  75.0  0.951076  1533.000000   1533.000000
Confusion Matrix:
[[1458    0]
 [  75    0]]
```

❄ مدل KNN

```
# Optimal K evaluation
optimal_k = k_values[np.argmax(accuracies)]
knn_optimal = KNeighborsClassifier(n_neighbors=optimal_k)
knn_optimal.fit(X_train, y_train)
y_pred_optimal = knn_optimal.predict(X_test)

# Report for optimal K
print(f"Optimal K: {optimal_k}")
print("Classification Report for Optimal K:")
print(classification_report(y_test, y_pred_optimal))
print("Confusion Matrix for Optimal K:")
print(confusion_matrix(y_test, y_pred_optimal))
```

بهینه سازی : از لیست accuracies مقدار K که بالاترین دقت را دارد، انتخاب می‌شود و این مقدار به عنوان K بهینه ذخیره می‌شود

سپس اطلاعات کسب شده از KNN را گزارش می دهیم که به صورت زیر است :

```
Optimal K: 6
Classification Report for Optimal K:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      1458
           1       0.00      0.00      0.00        75

    accuracy                           0.95      1533
   macro avg       0.48      0.50      0.49      1533
weighted avg       0.90      0.95      0.93      1533

Confusion Matrix for Optimal K:
[[1458    0]
 [  75    0]]
```

طبق گزارش بالا ، تعداد K بهینه برای نمایش دقت عالی ۶ تا می باشد

۱۶- ۱- اجرای الگوریتم ensemble model

برای اجرای یک الگوریتم Ensemble Model می‌توان از تکنیک‌های متداولی مانند Random Forest، Gradient Boosting، یا Voting Classifier استفاده کرد. در اینجا، روش Random Forest و Gradient Boosting (XGBoost) اجرا می‌شود، و نتایج آنها ارزیابی می‌شود.

۱۶- ۱- ۱- پیاده‌سازی Random Forest

Random Forest یک الگوریتم یادگیری ماشین است که برای مسائل دسته‌بندی (Classification) و رگرسیون (Regression) استفاده می‌شود. این الگوریتم و این با تنظیم وزن کلاس‌ها به صورت خودکار برای مقابله با داده‌های نامتوازن طراحی شده است.

```
# Initialize and train Random Forest
rf_model = RandomForestClassifier(random_state=42, class_weight="balanced")
rf_model.fit(X_train, y_train)
report_rf, matrix_rf = classification_report(y_test, rf_model.predict(X_test), output_dict=True), confusion_matrix(y_test, rf_model.predict(X_test))
```

RandomForestClassifier : کلاس اصلی برای اجرای الگوریتم Random Forest است.

random_state=42 : برای قابل تکرار بودن نتایج، از یک مقدار ثابت برای تصادفی‌سازی استفاده می‌شود.

class_weight="balanced" : وزن کلاس‌ها به‌صورت خودکار بر اساس معکوس توزیع کلاس‌ها تنظیم می‌شود

مدل Random Forestبا داده‌های آموزشی X_train و y_train آموزش داده می‌شود و مدل مجموعه‌ای از درخت‌های تصمیم‌گیری را ایجاد می‌کند.

classification_report : معیارهای عملکرد مدل را ارائه می‌دهد:

Precision: درصد نمونه‌هایی که به درستی به عنوان یک کلاس خاص پیش‌بینی شده‌اند.

Recall : درصد نمونه‌های واقعی یک کلاس که به درستی پیش‌بینی شده‌اند.

F1-Score : میانگین موزون Precision و Recall.

Support تعداد نمونه‌های واقعی در هر کلاس.

output_dict=True : خروجی به‌صورت یک دیکشنری ذخیره می‌شود که می‌توان آن را برای گزارش‌دهی دقیق‌تر استفاده کرد

confusion_matrix : تعداد پیش‌بینی‌های درست و نادرست را نشان می‌دهد : هر ردیف نمایانگر نمونه‌های واقعی و هر ستون نمایانگر پیش‌بینی‌های مدل.

۱۶- ۱- ۲- پیاده‌سازی XGBoost

```
# Initialize and train XGBoost
xgb_model = XGBClassifier(random_state=42, scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_train == 1]))
xgb_model.fit(X_train, y_train)
report_xgb, matrix_xgb = classification_report(y_test, xgb_model.predict(X_test), output_dict=True), confusion_matrix(y_test, xgb_model.predict(X_test))
```

یک الگوریتم قدرتمند یادگیری ماشین مبتنی بر روش بوستینگ Extreme Gradient Boosting مخفف عبارت XGBoost است. این الگوریتم برای مسائل دسته‌بندی و رگرسیون به‌طور گسترده استفاده می‌شود و به (Gradient Boosting) گرادیان می توان به بر پایه بوستینگ گرادیان و کارایی بالا و XGBoost دلیل سرعت و دقت بالایش شهرت زیادی دارد. ویژگی‌های اصلی . و تعامل با داده‌های گمشده اشاره کرد (Regularization) انعطاف‌پذیری و پشتیبانی از منظم‌سازی

بوستینگ گرادیان : از ترکیب چندین مدل ساده (مانند درخت‌های تصمیم) استفاده می‌کند تا مدل نهایی قوی‌تری بسازد . در هر .مرحله، مدل خطای مدل‌های قبلی را یاد می‌گیرد و آن را کاهش می‌دهد

کارایی بالا : بهینه‌سازی‌شده برای کار با مجموعه داده‌های بزرگ و پیاده‌سازی موازی و کارآمد برای کاهش زمان اجرا

.انعطاف‌پذیری : پشتیبانی از مسائل دسته‌بندی، رگرسیون، و رتبه‌بندی و قابل استفاده با داده‌های مختلف (پیوسته یا گسسته)

برای جلوگیری از بیش‌برازش L2 Regularization و L1 استفاده از : (Regularization) پشتیبانی از منظم‌سازی (Overfitting)

و تعامل با داده‌های گمشده : XGBoost به‌طور خودکار داده‌های گمشده را مدیریت می‌کند.

نحوه کار XGBoost

ایجاد درخت‌های تصمیم متوالی:

درخت‌های تصمیم کوچک به صورت متوالی ساخته می‌شوند.

هر درخت تلاش می‌کند خطای پیش‌بینی درخت‌های قبلی را کاهش دهد.

بوستینگ گرادیان:

خطای پیش‌بینی با استفاده از گرادیان تابع خطا (Loss Function) بهینه‌سازی می‌شود.

الگوریتم تلاش می‌کند مقدار خطا را در هر مرحله کاهش دهد.

وزن‌دهی نمونه‌ها:

نمونه‌هایی که سخت‌تر پیش‌بینی می‌شوند وزن بیشتری می‌گیرند تا مدل بعدی روی آن‌ها تمرکز کند.

۱۶- ۱- ۳- ارزیابی الگوریتم ensemble model

❄ مدل Random Forest

```
# Print results for Random Forest
print("Random Forest Results:")
print("Classification Report:")
print(pd.DataFrame(report_rf))
print("Confusion Matrix:")
print(matrix_rf)
```

مدل Random Forest خروجی زیر را به ما نمایش می دهد :

```
Random Forest Results:
Classification Report:
                     0     1  accuracy    macro avg  weighted avg
precision     0.951076   0.0  0.951076     0.475538      0.904546
recall        1.000000   0.0  0.951076     0.500000      0.951076
f1-score      0.974925   0.0  0.951076     0.487462      0.927228
support    1458.000000  75.0  0.951076  1533.000000   1533.000000
Confusion Matrix:
[[1458    0]
 [  75    0]]
```

❋ مدل XGBoost

```python
# Print results for XGBoost
print("\nXGBoost Results:")
print("Classification Report:")
print(pd.DataFrame(report_xgb))
print("Confusion Matrix:")
print(matrix_xgb)
```

مدل XGBoost خروجی زیر را به ما نمایش می دهد :

```
XGBoost Results:
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.959891   0.258065  0.931507     0.608978      0.925555
recall        0.968450   0.213333  0.931507     0.590892      0.931507
f1-score      0.964152   0.233577  0.931507     0.598864      0.928409
support    1458.000000  75.000000  0.931507  1533.000000   1533.000000
Confusion Matrix:
[[1412   46]
 [  59   16]]
```