مخزن گیت‌هاب

برای مشاهده و دانلود کدهای مربوط به این پروژه، می‌توانید به مخزن گیت‌هاب مراجعه کنید:

مخزن گیت‌هاب پروژه‌ها

دفترچه گوگل کولب

همچنین دفترچه‌ی گوگل کولب مربوط به تمرین را از لینک زیر مشاهده کنید:

AI4032_MP1 در Google Colab

گزارش مینی پروژه ۱
هوش مصنوعی

دکتر علیاری

سینا حسن‌پور شماره دانشجویی: ۴۰۲۱۶۷۲۳

۱۹ فروردین ۱۴۰۴

۱ پیش بینی بقا در کشتی تایتانیک با استفاده از رگرسیون لجستیک

مرحله اول: تحلیل اولیه داده‌ها (EDA)

در این مرحله ابتدا فایل CSV مربوط به داده‌های مسافران کشتی تایتانیک را در محیط گوگل کولب بارگذاری کردیم و با استفاده از کتابخانه Pandas آن را بررسی نمودیم. برای شروع، از توابع head() و info() برای مشاهده‌ی ساختار اولیه و نوع داده‌ها استفاده کردیم. سپس با تابع describe() آماری مانند میانگین، انحراف معیار، کمینه و بیشینه‌ی ویژگی‌های عددی را مشاهده کردیم.

```
import pandas as pd

# Read the dataset
df = pd.read_csv("titanic.csv")

# Show first rows of the data
df.head()

# Show structure of dataset
df.info()

# Describe numerical features
df.describe()
```

مشاهدات اولیه نشان داد که برخی ستون‌ها دارای مقادیر گمشده هستند (مثلاً ستون‌های Age و Cabin) و برخی ویژگی‌ها به صورت عددی و برخی به صورت متنی ثبت شده‌اند. ستون هدف Survived به صورت باینری (۰ و ۱) مشخص شده است که نشان‌دهنده‌ی مرگ یا نجات مسافران می‌باشد.
در ادامه، به تحلیل بصری و نمودارهای همبستگی خواهیم پرداخت تا بفهمیم کدام ویژگی‌ها بیشترین تأثیر را در بقا دارند.

تحلیل همبستگی ویژگی‌ها

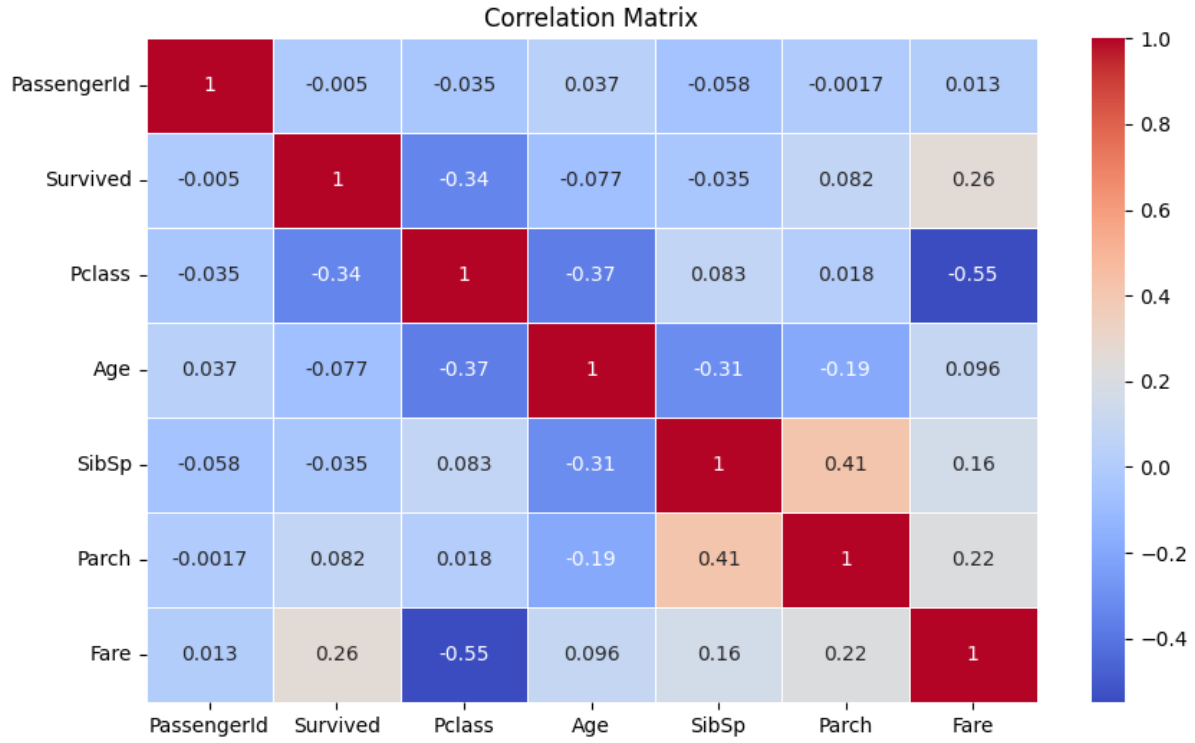
در این مرحله، هدف ما بررسی میزان همبستگی بین ویژگی‌های عددی موجود در داده‌ها با متغیر هدف Survived است. از آنجا که تابع corr() تنها قادر به محاسبه‌ی همبستگی میان ستون‌های عددی است، ابتدا با استفاده از select_dtypes فقط ستون‌های عددی را انتخاب کردیم. سپس با استفاده از seaborn.heatmap نمودار ماتریس همبستگی را رسم کردیم.

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numeric columns
numeric_df = df.select_dtypes(include=["number"])

# Compute correlation matrix
corr_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

با تحلیل ماتریس همبستگی مشاهده شد که:

- ویژگی Fare بیشترین همبستگی مثبت را با Survived دارد. این نشان می‌دهد افرادی که کرایه‌ی بیشتری پرداخت کرده‌اند (احتمالاً در کلاس بالاتری بوده‌اند)، شانس بیشتری برای بقا داشته‌اند.
- ویژگی Pclass بیشترین همبستگی منفی را با Survived دارد. این موضوع منطقی است زیرا مسافران کلاس پایین‌تر (مثلاً طبقه سوم) شانس کمتری برای نجات داشته‌اند.
- ویژگی‌های SibSp و Parch نیز همبستگی نسبتاً ضعیفی با بقا دارند.

در مراحل بعدی، ویژگی‌هایی مانند Sex و Embarked که غیرعددی هستند ولی اطلاعات مفیدی دارند، با استفاده از کدگذاری مناسب وارد مدل خواهند شد.



شکل ۱: نقشه حرارتی همبستگی بین ویژگی‌های عددی

تحلیل تأثیر جنسیت بر بقا

در طول اجرای پروژه، هنگام رسم نمودار همبستگی Correlation Matrix با استفاده از تابع corr() با خطایی مواجه شدم که نشان می‌داد داده‌های غیرعددی مثل ستون‌های Name، Sex و Ticket باعث مشکل در محاسبه شده‌اند. برای حل این مشکل، تصمیم گرفتم فقط ستون‌های عددی را جدا کنم و تحلیل را روی آن‌ها انجام دهم.

با این حال، پس از رفع خطا و مشاهده نتایج، به این فکر افتادم که برخی ستون‌های غیرعددی مانند Sex (جنسیت) ممکن است تأثیر بسیار زیادی در بقای مسافران داشته باشند. بنابراین تصمیم گرفتم این ویژگی مهم را به‌صورت جداگانه بررسی و تحلیل کنم.

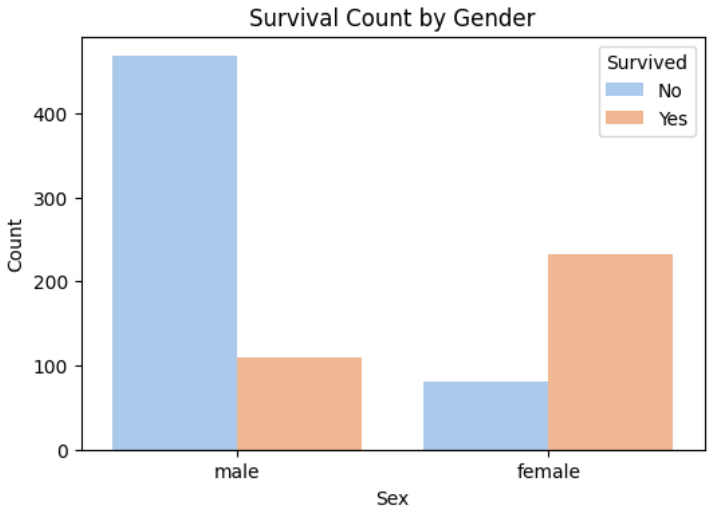
برای این کار، از نمودار countplot در کتابخانه‌ی Seaborn استفاده کردم تا رابطه‌ی بین جنسیت و بقا را به‌صورت بصری مشاهده کنم:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot count of survivors by gender
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Sex', hue='Survived', palette='pastel')
plt.title('Survival Count by Gender')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.legend(title='Survived', labels=['No', 'Yes'])
plt.show()
```

با مشاهده‌ی این نمودار، می‌توان نتیجه گرفت که:

- اکثر زنان مسافر نجات یافته‌اند.
- در مقابل، بیشتر مردان جان خود را از دست داده‌اند.

این تحلیل بصری نشان می‌دهد که ویژگی Sex تأثیر بسیار بالایی در تعیین بقای مسافران داشته و در مراحل بعدی باید حتماً در مدل‌سازی به آن توجه شود.



شکل ۲: نمودار تعداد بازماندگان و فوت‌شدگان بر اساس جنسیت

خلاصه کارهای انجام‌شده در این بخش

- با دستور select_dtypes فقط ستون‌های عددی را از داده‌ها جدا کردیم.

- با تابع df.corr() ماتریس همبستگی بین ویژگی‌ها را محاسبه کردیم.
- با استفاده از seaborn.heatmap() نمودار همبستگی را رسم نمودیم.
- تحلیل همبستگی‌ها به این صورت بود:
  - ویژگی Fare بیشترین همبستگی مثبت با متغیر Survived را داشت.
  - ویژگی Pclass بیشترین همبستگی منفی با Survived را نشان داد.
  - ویژگی‌های Age، SibSp و Parch نیز همبستگی ضعیف‌تری داشتند.

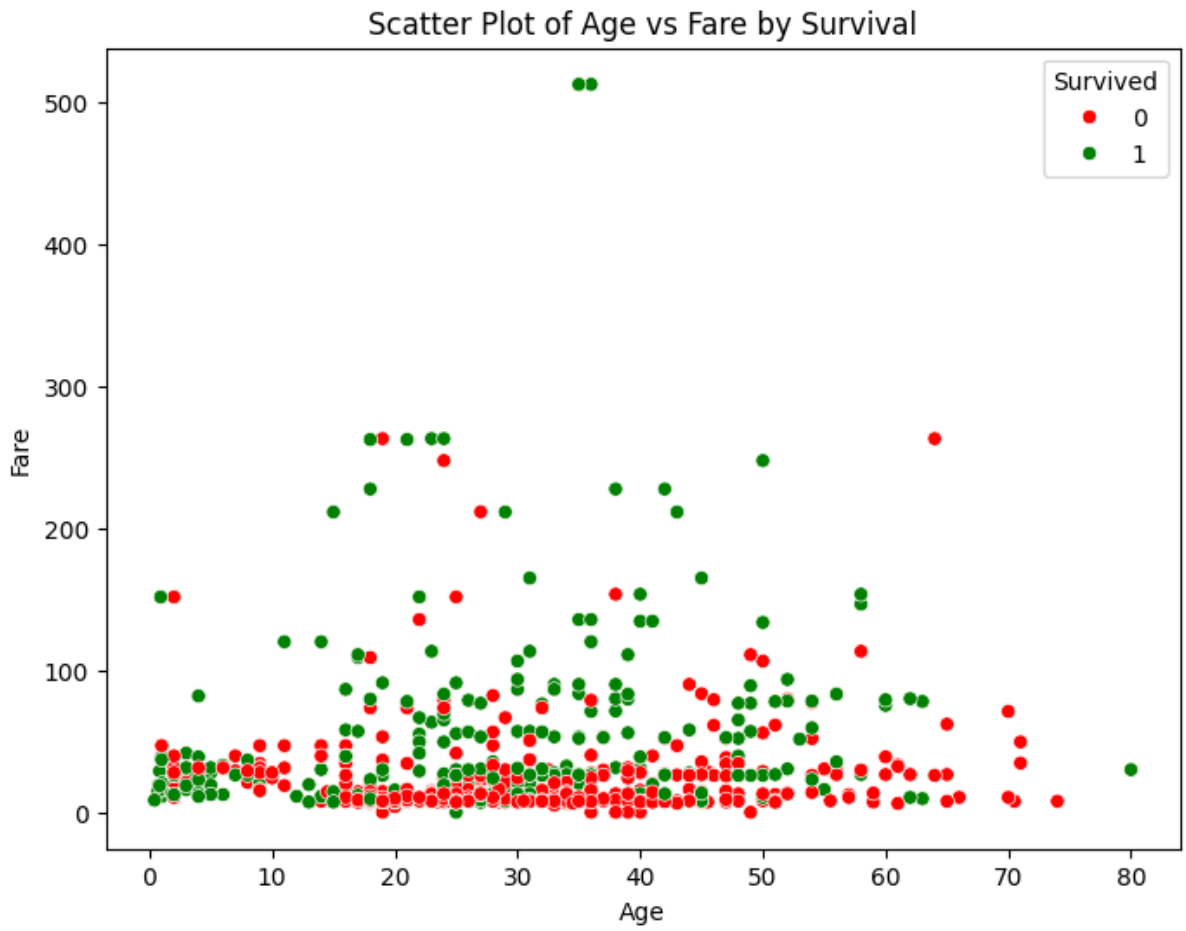
تحلیل بصری ویژگی‌های عددی Age و Fare با متغیر Survived

در این بخش، برای بررسی رابطه‌ی بین سن (Age) و کرایه‌ی پرداخت‌شده (Fare) با متغیر هدف (Survived)، از نمودار پراکندگی استفاده کردیم. در ابتدا با خطای رنگ‌بندی در راهنمای نمودار (Legend) مواجه شدیم که پس از بررسی، علت آن استفاده‌ی ناصحیح از برچسب‌های دستی بود. این مشکل با واگذاری مدیریت برچسب‌ها به کتابخانه‌ی Seaborn حل شد.
نمودار نهایی به‌گونه‌ای رسم شد که افراد فوت‌شده با رنگ قرمز و افراد نجات‌یافته با رنگ سبز نمایش داده شوند.

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.scatterplot(
    data=df,
    x='Age',
    y='Fare',
    hue='Survived',
    palette={0: 'red', 1: 'green'}
)
plt.title("Scatter Plot of Age vs Fare by Survival")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.legend(title="Survived", loc='best')
plt.savefig("Age_Fare_Scatter.png")
plt.show()
```

تحلیل نمودار نشان می‌دهد:

- تراکم افراد در سنین پایین تا متوسط و در بازه‌ی کرایه‌های پایین‌تر (زیر ۵۰) بیشتر است.
- در میان افراد با کرایه‌های بالا (بیش از ۱۰۰)، نسبت بازماندگان بیشتر از فوت‌شدگان است.
- سنین خیلی پایین یا خیلی بالا، الگوی خاصی برای بقا نشان نمی‌دهند؛ اما افراد بین ۱۵ تا ۴۰ ساله با کرایه‌ی بالا، بخت بیشتری برای نجات داشته‌اند.



شکل ۳: نمودار پراکندگی سن و کرایه با توجه به بقای مسافران

نمودار هگزبین برای بررسی چگالی سنی و کرایه

نمودار پراکندگی به ما امکان مشاهده‌ی مستقیم هر داده را می‌دهد، اما در شرایطی که تعداد داده‌ها زیاد باشد، نقاط روی هم افتاده و تحلیل دشوار می‌شود. برای حل این مشکل، از نمودار Hexbin استفاده کردیم.

در این نمودار، فضا به شش‌ضلعی‌های هم‌اندازه تقسیم می‌شود و تعداد نقاط موجود در هر سلول با شدت رنگ نمایش داده می‌شود. این روش به خوبی چگالی داده‌ها را در نواحی مختلف نشان می‌دهد.

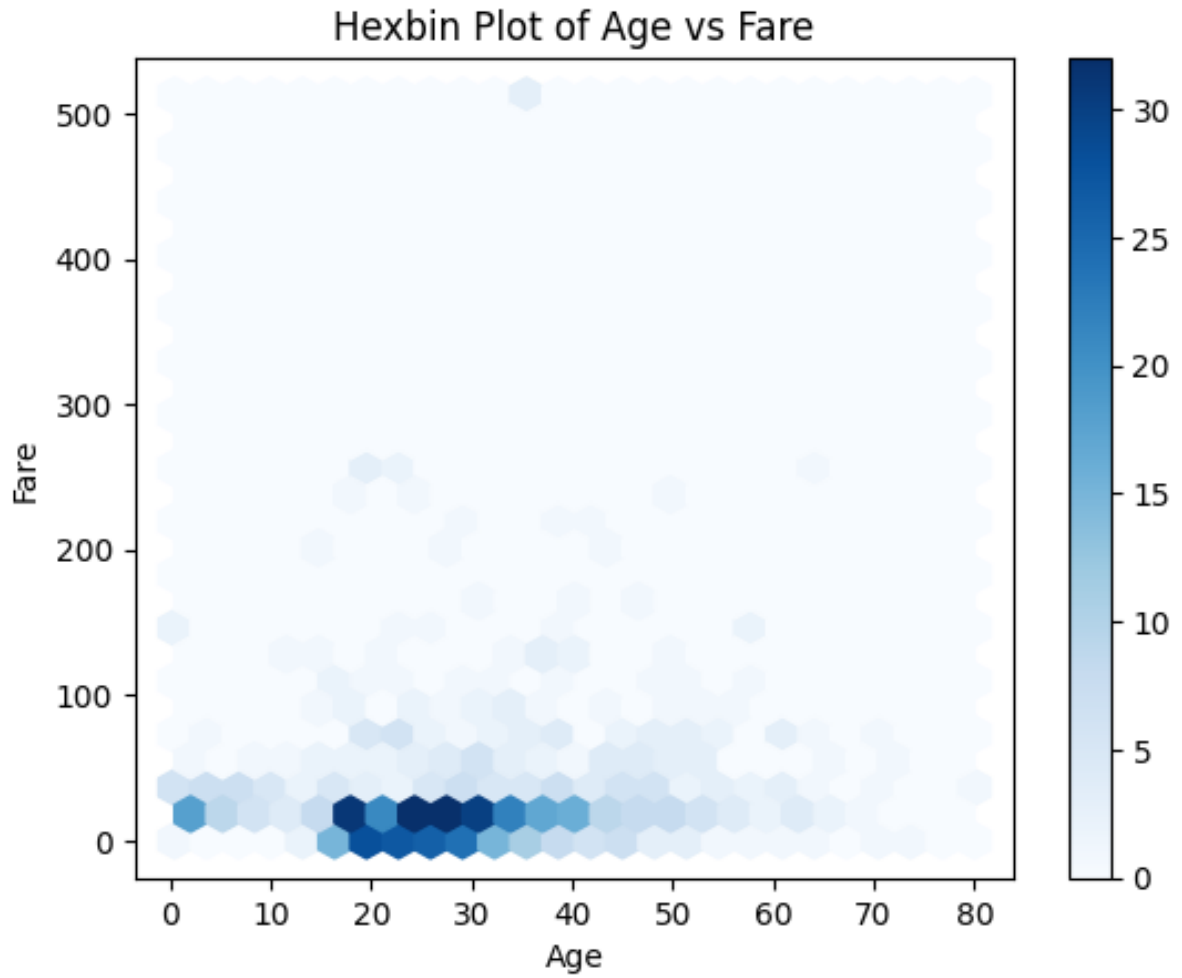
برای رسم این نمودار، ابتدا باید مقادیر گمشده‌ی ستون‌های Age و Fare را حذف کنیم تا تابع hexbin به درستی اجرا شود.

```
# Hexbin plot (requires dropna)
plt.figure(figsize=(8,6))
df.dropna(subset=['Age', 'Fare']).plot.hexbin(
    x='Age', y='Fare', gridsize=25, cmap='Blues', sharex=False,
        sharey=False)
plt.title("Hexbin Plot of Age vs Fare")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.show()
```

از نمودار هگزبین نتایج زیر قابل برداشت است:

- بیشترین تراکم داده‌ها در بازه‌ی سنی حدود ۲۰ تا ۴۰ سال و کرایه‌ی کمتر از ۵۰ دیده می‌شود.
- نقاطی با کرایه‌ی بسیار بالا (بیش از ۲۰۰) و سن متوسط، به صورت پراکنده و کم‌تعداد ظاهر شده‌اند.

- این نمودار کمک می‌کند تا نواحی پرتراکم بدون شلوغی بیش‌ازحد قابل تشخیص باشند.



شکل ۴: نمودار هگزبین برای سن و کرایه در مجموعه داده تایتانیک

خلاصه‌ی کاربرد نمودارهای پراکندگی و هگزبین

نمودارهای (Scatter) و (Hexbin) ابزارهای بسیار مفیدی برای تحلیل دو ویژگی عددی به صورت همزمان هستند. این نمودارها اطلاعات زیر را در اختیار ما قرار می‌دهند:

- بررسی توزیع داده‌ها در فضای دو بعدی (مانند سن و کرایه)
- مشاهده‌ی الگوهای احتمالی میان دو ویژگی عددی
- تشخیص روابط غیرخطی یا خوشه‌بندی‌های پنهان در داده‌ها
- تحلیل چگالی داده‌ها در نواحی مختلف (به‌ویژه در نمودار هگزبین)

در پروژه‌ی تایتانیک، از این نمودارها برای تحلیل رابطه‌ی سن و کرایه با احتمال بقا استفاده شد. این نمودارها نشان دادند که بسیاری از بازماندگان در بازه‌ی سنی خاصی و با کرایه‌های بالاتر قرار داشتند. همچنین تراکم فوت‌شدگان در نواحی خاصی بیشتر بود که می‌تواند در مدل‌سازی نهایی مؤثر باشد.

تحلیل توزیع بازماندگان بر اساس سن، کرایه پرداختی و جنسیت

در این بخش، به بررسی سه جنبه مهم از داده‌های مسافران کشتی تایتانیک پرداختیم:

- بررسی توزیع بازماندگان بر اساس سن و کرایه پرداختی با استفاده از نمودار تعاملی Plotly
- بررسی درصد نجات‌یافتگان در میان مردان و زنان
- مقایسه‌ی توزیع بازماندگان به تفکیک جنسیت با استفاده از countplot

تحلیل توزیع بازماندگان بر اساس سن، کرایه پرداختی و جنسیت

در این بخش، به‌منظور تحلیل دقیق‌تر متغیر هدف (Survived)، سه بررسی مختلف انجام دادیم:

۱. رسم نمودار پراکندگی بازماندگان بر اساس سن و کرایه پرداختی با استفاده از Plotly
۲. بررسی تأثیر میزان کرایه بر شانس بقا
۳. محاسبه درصد بازماندگان به تفکیک جنسیت و نمایش گرافیکی با Seaborn

۱. نمودار پراکندگی Plotly: بررسی سن و کرایه در ارتباط با بقا

برای بررسی ارتباط بین سن، کرایه و بقا، از نمودار پراکندگی تعاملی استفاده کردیم که در آن:

- محور افقی نمایانگر سن (Age)
- محور عمودی نمایانگر کرایه پرداختی (Fare)
- رنگ نقاط تعیین‌شده بر اساس وضعیت بقا (Survived) — قرمز برای فوت‌شده، سبز برای بازمانده

```python
import plotly.express as px

filtered_df = df.dropna(subset=['Age', 'Fare'])

fig = px.scatter(
    filtered_df,
    x="Age",
    y="Fare",
    color="Survived",
    color_continuous_scale=["red", "green"],
    title="Age vs Fare Scatter Plot Colored by Survival",
    labels={"Survived": "Survived"},
    hover_data=["Sex", "Pclass"]
)
fig.show()
```

تحلیل:

- افرادی که کرایه‌ی بالاتری پرداخت کرده‌اند، معمولاً در طبقات بالاتر قرار داشتند و به همین دلیل شانس بقای بیشتری داشتند.
- تمرکز بازماندگان در بازه‌های خاصی از سن و کرایه مشاهده شد (مثلاً سنین ۲۰ تا ۴۰ و کرایه‌های بالای ۵۰).

۲. بررسی آماری درصد بازماندگان به تفکیک جنسیت

برای پاسخ به این سؤال که "چه درصدی از زنان و چه درصدی از مردان نجات یافته‌اند"، ابتدا با استفاده از توابع آماری، تعداد بازماندگان هر جنس را به نسبت کل افراد آن جنس محاسبه کردیم:

```
# Survival percentage by gender
survived_counts = df[df['Survived'] == 1]['Sex'].value_counts()
total_counts = df['Sex'].value_counts()
survival_percent = (survived_counts / total_counts) * 100
print(survival_percent)
```

نتایج محاسبه:

- درصد بازماندگان زن: 74.2%
- درصد بازماندگان مرد: 18.9%

این اختلاف قابل توجه نشان می‌دهد که جنسیت نقش بسیار مؤثری در احتمال بقا داشته است.

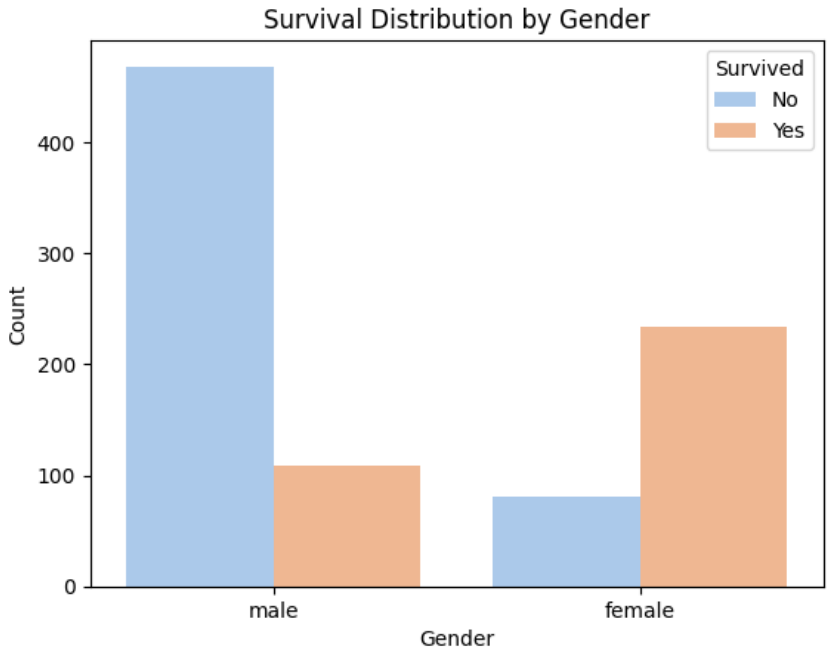
۳. نمایش گرافیکی توزیع بازماندگان با Seaborn

در ادامه، برای نمایش بصری بهتر توزیع بقا بین مردان و زنان، از نمودار $\text{countplot}()$ استفاده کردیم:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(data=df, x='Sex', hue='Survived', palette='pastel')
plt.title("Survival Count by Gender")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.legend(title='Survived', labels=['No', 'Yes'])
plt.savefig("Survival_by_Gender.png")
plt.show()
```

تحلیل نمودار:

- بیشترین فوت‌شدگان از میان مردان بوده‌اند.
- اکثر زنان موفق به نجات شده‌اند.



شکل ۵: نمودار توزیع بازماندگان و فوت‌شدگان به تفکیک جنسیت

تحلیل آماری: بررسی تأثیر خانواده و سن بر احتمال بقا

در این بخش، سه تحلیل آماری برای بررسی عوامل تأثیرگذار بر بقا انجام دادیم:

۱. بررسی تأثیر تعداد اعضای خانواده همراه با مسافر

۲. بررسی تفاوت در بقا بین افراد تنها و افرادی که با خانواده سفر کرده‌اند

۳. تحلیل شانس بقا در گروه‌های مختلف سنی

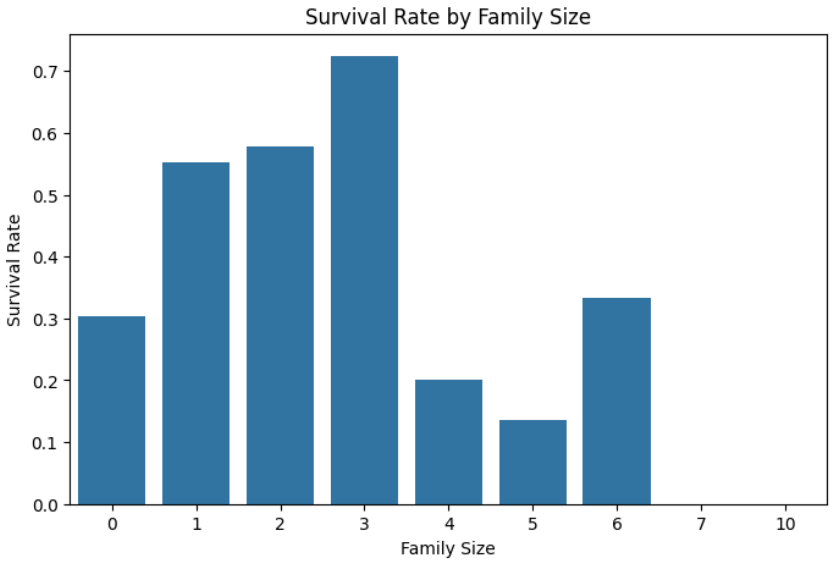
۱. تأثیر تعداد اعضای خانواده (FamilySize)

ابتدا با جمع ستون‌های SibSp و Parch، ستون جدیدی به نام FamilySize تعریف کردیم که نشان می‌دهد هر مسافر همراه با چند نفر از خانواده‌اش در کشتی حضور داشته است.
سپس با استفاده از نمودار میله‌ای، میانگین نرخ بقا برای اندازه‌های مختلف خانواده را بررسی کردیم:

```
df["FamilySize"] = df["SibSp"] + df["Parch"]

sns.barplot(data=df, x="FamilySize", y="Survived", ci=None)
```

تحلیل:

- افرادی که خانواده‌ی کوچکی (۱ تا ۳ نفر) همراه داشتند، شانس بقا بیشتری داشتند.
- افرادی که تنها بودند یا خانواده‌ی خیلی بزرگی داشتند (بیش از ۵ نفر)، نرخ بقای پایین‌تری داشتند.



شکل ۶: نرخ بقا بر اساس تعداد اعضای خانواده

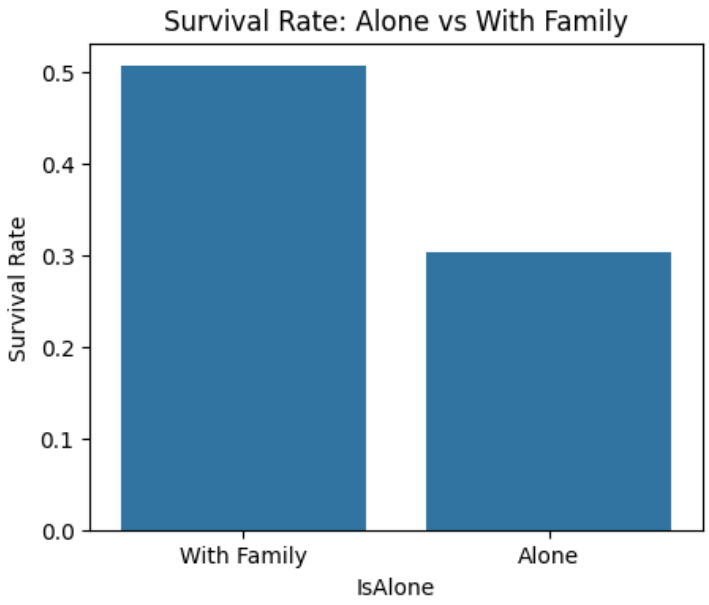
۲. بررسی تنها بودن (IsAlone)

برای تحلیل اثر تنهایی بر بقا، ستونی به نام IsAlone تعریف کردیم که مشخص می‌کند آیا هر مسافر تنها بوده یا نه.

```
df["IsAlone"] = (df["FamilySize"] == 0).astype(int)

sns.barplot(data=df, x="IsAlone", y="Survived", ci=None)
```

تحلیل:

- افرادی که همراه با خانواده سفر کرده‌اند، نرخ بقای بیشتری داشتند.
- به‌نظر می‌رسد تنهایی عاملی منفی در بقا بوده است.



شکل ۷: نرخ بقا بر اساس تنها بودن یا همراه داشتن خانواده

۳. تحلیل گروه‌های سنی (AgeGroup)

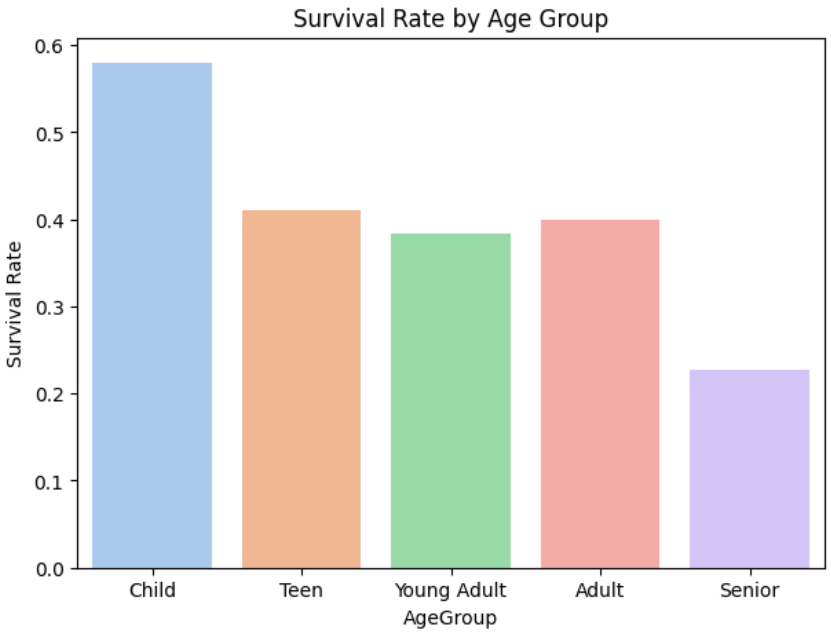
برای بررسی تأثیر سن، ابتدا داده‌ها را به گروه‌های سنی تقسیم کردیم:

- Child (۰ تا ۱۲ سال)
- Teen (۱۳ تا ۱۹ سال)
- Young Adult (۲۰ تا ۳۵ سال)
- Adult (۳۶ تا ۶۰ سال)
- Senior (بیش از ۶۰ سال)

```
bins = [0, 12, 19, 35, 60, 100]
labels = ['Child', 'Teen', 'Young Adult', 'Adult', 'Senior']
df["AgeGroup"] = pd.cut(df["Age"], bins=bins, labels=labels)

sns.barplot(data=df, x="AgeGroup", y="Survived", ci=None)
```

تحلیل:

- کودکان و نوجوانان شانس بقای نسبتاً بالایی داشتند.
- نرخ بقا در میان سالمندان به شکل قابل توجهی پایین‌تر بود.
- به‌طور کلی، افراد جوان و همراه با خانواده، بیشترین شانس نجات را داشتند.



شکل ۸: نرخ بقا بر اساس گروه‌های سنی

بخش دوم: پیش‌پردازش مجموعه داده

در این بخش، مجموعه داده‌های خام مربوط به مسافران کشتی تایتانیک، برای استفاده در مدل‌های یادگیری ماشین آماده‌سازی شدند. مراحل زیر با دقت و بررسی وجود ستون‌ها انجام شد:

۱. بارگذاری داده و بررسی اولیه

ابتدا فایل Titanic-Dataset.csv در محیط Colab Google بارگذاری و با تابع df.info() بررسی شد. خروجی نشان داد که داده‌ها شامل ۸۹۱ ردیف هستند.

۲. حذف ویژگی‌های غیرضروری

ویژگی‌هایی مانند Name، Ticket، Cabin و PassengerId اطلاعات مفیدی برای مدل‌سازی نداشتند و حذف شدند:

```
columns_to_drop = ["Name", "Ticket", "Cabin", "PassengerId"]
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns])
```

۳. تبدیل داده‌های متنی به عددی

- ستون Sex به مقدار عددی ۰ (مرد) و ۱ (زن) نگاشت شد.
- ستون Embarked با مقدار پرتکرار (mode) پر شده و سپس با کدگذاری One-Hot به دو ستون Embarked_Q و Embarked_S تبدیل شد.

```
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})

df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])
df = pd.get_dummies(df, columns=["Embarked"], drop_first=True)
```

۴. پر کردن مقادیر گمشده

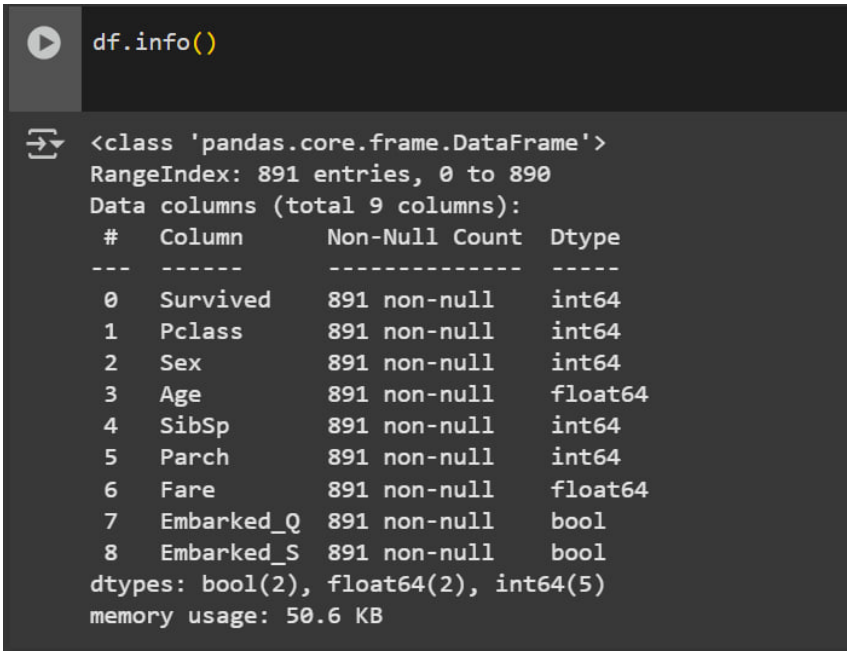
برای جلوگیری از حذف داده‌های مفید، مقادیر گمشده با میانگین یا میانه پر شدند:

```
df["Age"] = df["Age"].fillna(df["Age"].median())
df["Fare"] = df["Fare"].fillna(df["Fare"].median())
```

۵. نتیجه نهایی

در نهایت، با اجرای df.info() مشاهده شد که:

- هیچ داده‌ی گمشده‌ای باقی نمانده است.
- تمام ستون‌ها عددی یا بولی هستند.
- داده‌ها آماده‌ی ورود به مرحله‌ی آموزش مدل هستند.



شکل ۹: خروجی کد

پاسخ به سوالات بخش دوم پیش‌پردازش مجموعه داده

پیش‌پردازش داده‌ها یکی از حیاتی‌ترین مراحل در پروژه‌های یادگیری ماشین است. در این مرحله تلاش می‌کنیم داده‌های خام را تمیز، ساخت‌یافته و آماده‌ی مدل‌سازی کنیم تا یادگیری مدل مؤثر و دقیق باشد. در ادامه، به سؤالات این بخش به‌صورت گام‌به‌گام و تحلیلی پاسخ داده‌ایم.

سؤال ۱: بررسی داده‌های گمشده و روش‌های پر کردن آن‌ها

ابتدا با استفاده از دستور زیر، ستون‌هایی که دارای داده‌های گمشده هستند را بررسی کردیم:

```
df.isnull().sum()
```

نتایج:

- ستون Age دارای داده‌های گمشده بود.
- ستون Embarked چند مقدار گمشده داشت.
- ستون Cabin بیش از ۷۰٪ داده‌ی گمشده داشت.

سه روش برای مدیریت داده‌های گمشده استفاده کردیم:

۱. **میانه (Median)** برای ستون Age:

```
df["Age"] = df["Age"].fillna(df["Age"].median())
```

۲. **پرتکرار (Mode)** برای ستون Embarked:

```
df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])
```

۳. **حذف ستون با داده‌ی بسیار گمشده** برای ستون Cabin:

```
df.drop("Cabin", axis=1, inplace=True)
```

سؤال ۲: حذف ستون‌های غیرضروری

برخی از ستون‌ها اطلاعات مفیدی برای مدل‌سازی ارائه نمی‌دهند، بنابراین حذف شدند:

ستون	دلیل حذف
Name	شامل نام افراد است و به‌صورت مستقیم قابل استفاده در مدل نیست
Ticket	شامل کدهای نامنظم بلیت است که معنای خاصی ندارند
PassengerId	فقط شناسه رکورد است و تأثیری در بقا ندارد
Cabin	دارای مقادیر گمشده‌ی زیاد بود (بیش از ۷۰٪)

جدول ۱: ستون‌های حذف‌شده از مجموعه داده

```
columns_to_drop = ["Name", "Ticket", "Cabin", "PassengerId"]
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns])
```

سؤال ۳: ویژگی‌های عددی و دسته‌ای

- **ویژگی‌های عددی**: شامل مقادیر عددی قابل اندازه‌گیری هستند. مثال: Age، Fare، SibSp، Parch
- **ویژگی‌های دسته‌ای**: شامل مقادیر گسسته و متنی هستند. مثال: Sex، Embarked، Pclass

سؤال ۴: پیش‌پردازش ویژگی‌های دسته‌ای

برای استفاده از ویژگی‌های دسته‌ای در مدل، لازم است آن‌ها را به صورت عددی تبدیل کنیم:

- ستون Sex به صورت عددی با نگاشت ۰ و ۱ تبدیل شد:

```
    df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
```

- ستون Embarked با استفاده از One-Hot Encoding به ستون‌های Embarked_Q و Embarked_S تبدیل شد:

```
    df = pd.get_dummies(df, columns=["Embarked"], drop_first=True)
```

سؤال ۵: نرمال‌سازی و استانداردسازی

در یادگیری ماشین، نرمال‌سازی و استانداردسازی معمولاً در ویژگی‌های عددی به کار می‌روند:

- **نرمال‌سازی (Min-Max Scaling)**: تبدیل داده‌ها به بازه‌ی [۰،۱]
- **استانداردسازی (Standardization)**: تبدیل داده‌ها به میانگین صفر و واریانس یکتبدیل داده‌ها به میانگین صفر و واریانس یک

در این پروژه: چون از مدل لجستیک رگرسیون استفاده شده و اختلاف مقیاس بین ویژگی‌ها زیاد نیست، استفاده از این تکنیک‌ها ضروری نبود. اما در الگوریتم‌هایی مانند KNN یا SVM، این مرحله اهمیت بیشتری دارد.

بخش سوم: انتخاب ویژگی، آموزش و ارزیابی مدل

انتخاب ویژگی‌ها

در این مرحله، ابتدا ویژگی هدف Survived که نشان‌دهنده‌ی وضعیت بقا یا مرگ مسافران است، به‌عنوان متغیر y انتخاب شد. سپس سایر ویژگی‌ها به‌عنوان ورودی مدل (X) در نظر گرفته شدند.

```
#
y = df["Survived"]

#              )   Survived)
X = df.drop("Survived", axis=1)

#
X.shape, y.shape
```

نتیجه‌ی اجرای X.shape و y.shape نشان داد که مجموعه داده شامل:

- **۸۹۱ نمونه** (مسافر)
- **۸ ویژگی ورودی** برای هر مسافر

در گام بعد، با استفاده از مدل LogisticRegression به آموزش مدل خواهیم پرداخت و عملکرد آن را ارزیابی خواهیم کرد.

آموزش و ارزیابی مدل Regression Logistic

در این مرحله، داده‌ها را به دو بخش آموزش و تست با نسبت ۸۰٪ - ۲۰٪ تقسیم کردیم. سپس مدل LogisticRegression از کتابخانه scikit-learn را آموزش دادیم و بر روی داده‌ی تست ارزیابی کردیم.

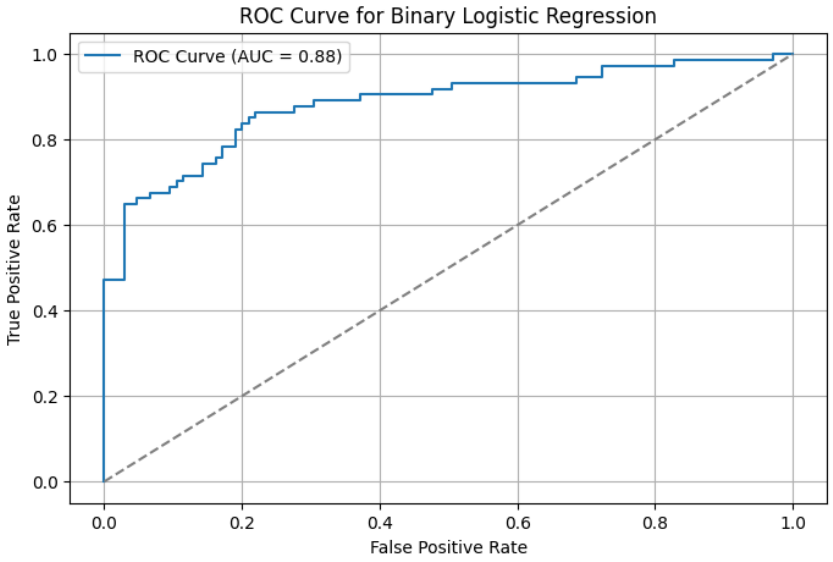
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report

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

#
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

#
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_mat = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_mat)
print("Classification Report:\n", report)
```

نتایج ارزیابی مدل

- دقت نهایی مدل (Accuracy) : ۸۱%
- مدل توانست با دقت خوبی کلاس‌های بقا (۱) و مرگ (۰) را پیش‌بینی کند.
- دقت (Precision) برای نجات‌یافته‌ها (کلاس ۱): 79%
- Recall برای نجات‌یافته‌ها: 74% — که نشان می‌دهد تعداد محدودی از بازماندگان به اشتباه فوت‌شده پیش‌بینی شده‌اند.
- F1-Score برای نجات‌یافته‌ها برابر ۷۶% است که نشان‌دهنده‌ی عملکرد متعادل مدل می‌باشد.

جدول ۲: ماتریس آشفتگی مدل لجستیک رگرسیون

	پیش‌بینی = ۰	پیش‌بینی = ۱
واقعاً ۰	۹۰	۱۵
واقعاً ۱	۱۹	۵۵



شکل ۱۰: شکل: نمودار ROC مدل لجستیک دودویی با AUC برابر (0.88)

تحلیل نمودار ROC

نمودار ROC عملکرد مدل را در تشخیص بازماندگان از غیر بازماندگان به تصویر می‌کشد. مدل ما دارای AUC برابر با (0.88) است که نشان‌دهنده‌ی قدرت تفکیک بسیار مناسب مدل می‌باشد. مقدار AUC بالا به این معناست که مدل می‌تواند به‌خوبی میان کلاس‌های مثبت و منفی تمایز قائل شود.

تحلیل ویژگی‌ها با استفاده از ضرایب مدل

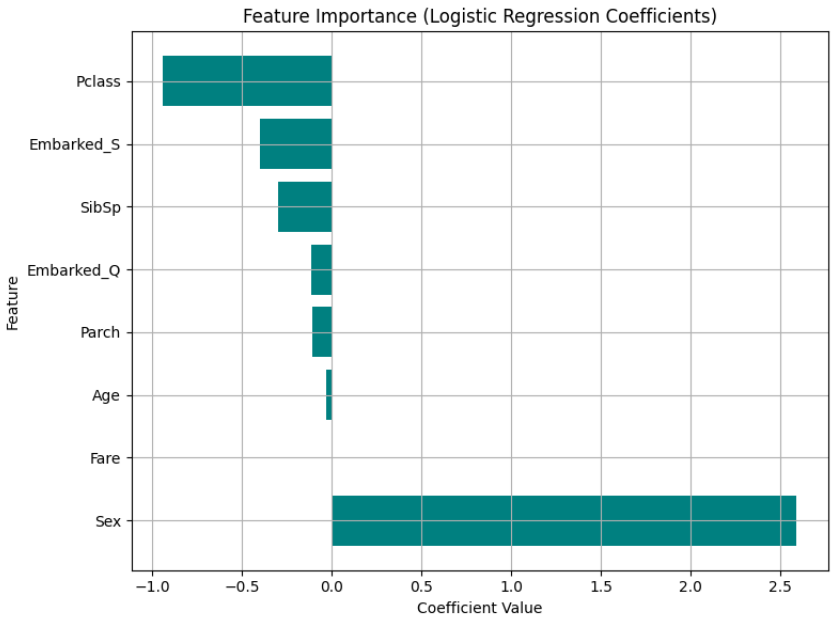
پس از آموزش مدل Logistic Regression، برای بررسی اهمیت ویژگی‌ها از ضرایب مدل استفاده کردیم. هر ضریب بیانگر تأثیر مستقیم آن ویژگی بر احتمال بقا است. اگر ضریب مثبت باشد، افزایش آن ویژگی احتمال بقا را افزایش می‌دهد و بالعکس.

```
coefficients = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": model.coef_[0]
})
coefficients.sort_values(by="Coefficient", ascending=False, inplace=True)
coefficients
```

نتایج تحلیل ضرایب

- **جنسیت (Sex)**: دارای بزرگ‌ترین ضریب مثبت $(+2.59)$ است. این نشان می‌دهد که زن بودن بیشترین تأثیر مثبت در شانس بقا دارد.
- **کلاس مسافرت (Pclass)**: ضریب 0.93- دارد که نشان‌دهنده تأثیر منفی شدید است. مسافران کلاس پایین‌تر شانس بقا کمتری دارند.
- **سن (Age)**: با ضریب منفی 0.03- تأثیر کمی در کاهش احتمال بقا دارد.
- **کرایه (Fare)**: ضریب مثبت اما کوچک $+0.0025$ دارد که نشان می‌دهد افراد ثروتمند اندکی شانس بیشتری دارند.
- سایر ویژگی‌ها مانند SibSp، Parch و اسکله‌های سوار شدن نیز تأثیرات جزئی دارند.

در تصویر زیر، نمودار ضرایب به ترتیب اهمیت نمایش داده شده است:



شکل ۱۱: نمودار اهمیت ویژگی‌ها در مدل لجستیک رگرسیون

انتخاب ویژگی با روش‌های مختلف

برای افزایش دقت مدل و کاهش پیچیدگی، از دو روش مختلف انتخاب ویژگی استفاده کردیم:

۱. روش Regression Lasso

در این روش از مدل Logistic RegressionCV با پنالتی L1 استفاده شد. این مدل می‌تواند ویژگی‌هایی با اهمیت کم را به صفر رسانده و حذف کند.

```
#       Lasso
lasso = LogisticRegressionCV(cv=5, penalty='l1', solver='liblinear')
lasso.fit(X_scaled, y)
selected_features = X.columns[lasso.coef_[0] != 0]
```

در نتیجه‌ی این روش، هیچ‌کدام از ویژگی‌ها حذف نشدند و تمام ۸ ویژگی زیر انتخاب شدند:
['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_Q', 'Embarked_S']

۲. روش RFE - حذف بازگشتی ویژگی‌ها

در این روش از Recursive Feature Elimination با مدل پایه لجستیک استفاده شد. این الگوریتم به صورت تدریجی ویژگی‌هایی که کمترین تأثیر را دارند حذف می‌کند تا به تعداد مشخصی برسد (در اینجا ۵ ویژگی).

```
#     RFE
rfe = RFE(LogisticRegression(), n_features_to_select=5)
rfe.fit(X, y)
selected_features = X.columns[rfe.support_]
```

ویژگی‌های انتخاب‌شده توسط RFE عبارت‌اند از:
['Pclass', 'Sex', 'SibSp', 'Embarked_Q', 'Embarked_S']

مقایسه دو روش

- روش **Lasso** هیچ ویژگی‌ای را حذف نکرد، زیرا تمام متغیرها در مدل مفید بودند.
- روش **RFE** با هدف کاهش ابعاد، ۵ ویژگی مهم‌تر را نگه داشت.

در ادامه مدل نهایی را بر اساس هر دو روش آموزش داده و عملکرد آن‌ها را مقایسه خواهیم کرد.

مقایسه عملکرد مدل براساس انتخاب ویژگی

در این بخش، مدل لجستیک رگرسیون را با دو مجموعه ویژگی آموزش دادیم: یک‌بار براساس ویژگی‌های انتخاب‌شده توسط روش Lasso Regression و بار دیگر با استفاده از ویژگی‌های منتخب روش Recursive Feature Elimination (RFE). در ادامه نتایج به‌دست آمده را با هم مقایسه می‌کنیم.

۱. مدل آموزش‌دیده با ویژگی‌های Lasso

- دقت مدل (Accuracy): **۸۱%**
- Precision برای بازماندگان (کلاس ۱): **۷۹%**
- Recall برای بازماندگان: **۷۴%**
- F1-score برای بازماندگان: **۷۶%**

Confusion Matrix (Lasso):

	Pred = 0	Pred = 1
True = 0	90	15
True = 1	19	55

۲. مدل آموزش‌دیده با ویژگی‌های RFE

- دقت مدل (Accuracy) : **۷۷%**
- Precision برای بازماندگان (کلاس ۱): **۷۱%**
- Recall برای بازماندگان: **۷۶%**
- F1-score برای بازماندگان: **۷۳%**

Confusion Matrix (RFE):

	Pred = 0	Pred = 1
True = 0	82	23
True = 1	18	56

تحلیل نهایی

- مدل مبتنی بر Lasso عملکرد بهتری در دقت کلی و میانگین معیارهای ارزیابی داشت.
- با اینکه مدل RFE ویژگی‌های کمتری داشت و ساده‌تر بود، اما عملکرد آن کمی ضعیف‌تر بود.
- انتخاب ویژگی با Lasso برای این مسئله خاص مناسب‌تر به‌نظر می‌رسد.

تبدیل متغیر هدف به سه کلاس و آموزش مدل لجستیک چندکلاسه

در این مرحله، با استفاده از احتمال پیش‌بینی‌شده بقا، متغیر هدف را به سه کلاس مجزا تقسیم کردیم:

- **کلاس ۰ (Low Chance)** : احتمال بقا کمتر از 0.33
- **کلاس ۱ (Medium Chance)** : احتمال بقا بین 0.33 تا 0.66
- **کلاس ۲ (High Chance)** : احتمال بقا بیشتر از 0.66

پس از تعیین این سه دسته، یک مدل Logistic Regression با رویکرد multinomial برای پیش‌بینی این کلاس‌ها آموزش داده شد.

توزیع نمونه‌ها در هر کلاس:

- کلاس ۰ (Low) : ۴۹۰ نمونه
- کلاس ۱ (Medium) : ۱۷۸ نمونه
- کلاس ۲ (High) : ۲۲۳ نمونه

نتایج مدل لجستیک چندکلاسه بر روی داده‌های تست:

- **دقت نهایی مدل (Accuracy)**: ٪۹۶
- **Precision برای کلاس‌ها:**
  - کلاس ۰: ٪۹۸

- کلاس ۱: ٪۹۱
- کلاس ۲: ٪۹۶

- **Recall برای کلاس‌ها:**
  - کلاس ۰: ٪۹۹
  - کلاس ۱: ٪۸۹
  - کلاس ۲: ٪۹۶
- **F۱-score برای کلاس‌ها:**
  - کلاس ۰: ٪۹۸
  - کلاس ۱: ٪۹۰
  - کلاس ۲: ٪۹۶

ماتریس آشفتگی مدل لجستیک چندکلاسه:

	پیش‌بینی: ۰	پیش‌بینی: ۱	پیش‌بینی: ۲
واقعاً ۰	۹۰	۱	۰
واقعاً ۱	۲	۳۲	۲
واقعاً ۲	۰	۲	۵۰

جدول ۳: ماتریس آشفتگی مدل لجستیک چندکلاسه

کلاس	Precision	Recall	F۱-score	Support
(۰) Low	0.98	0.99	0.98	91
(۱) Medium	0.91	0.89	0.90	36
(۲) High	0.96	0.96	0.96	52
avg Macro	0.95	0.95	0.95	179
avg Weighted	0.96	0.96	0.96	179

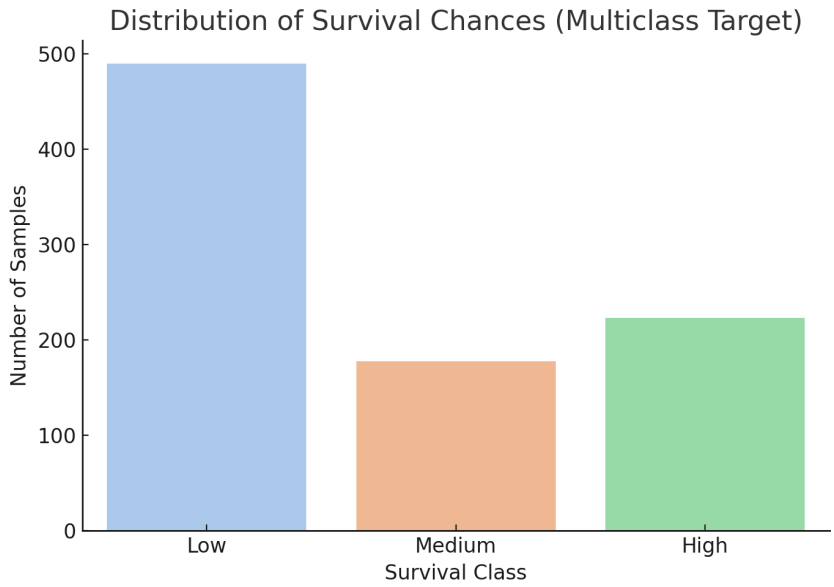
جدول ۴: گزارش عملکرد مدل لجستیک چندکلاسه

مدل لجستیک رگرسیون چندکلاسه برای پیش‌بینی احتمال بقا

در این بخش، متغیر هدف (Survived) را به سه کلاس احتمال بقا تقسیم کرده بودیم برای این کار، ابتدا مدل لجستیک رگرسیون دودویی را آموزش دادیم تا احتمال بقا برای هر مسافر پیش‌بینی شود. سپس با استفاده از تابع pd.cut، احتمال‌ها را به سه دسته‌ی بالا تقسیم کردیم. در ادامه، مدل لجستیک رگرسیون چندکلاسه را آموزش داده و ارزیابی کردیم.

توزیع کلاس‌های جدید

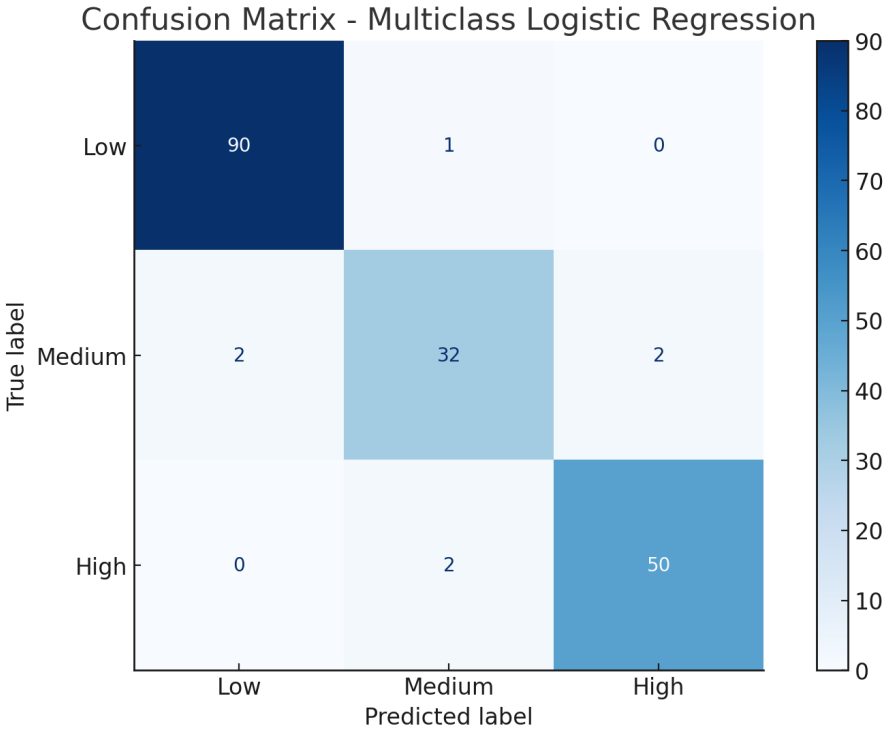
با رسم نمودار میله‌ای، توزیع نمونه‌ها در سه کلاس به‌صورت زیر مشاهده شد:



شکل ۱۲: نمودار توزیع تعداد نمونه در کلاس‌های Low ، Medium و High

ماتریس آشفتگی و گزارش عملکرد مدل

مدل نهایی با دقت ۹۶%، عملکرد بسیار خوبی در پیش‌بینی کلاس‌های سه‌گانه نشان داده است. در تصویر زیر، ماتریس آشفتگی (Confusion Matrix) و گزارش کامل از دقت، یادآوری (Recall) و F۱-Score برای هر کلاس آمده است:



شکل ۱۳: گزارش ارزیابی مدل لجستیک رگرسیون چندکلاسه

تحلیل نتایج

- مدل در پیش‌بینی کلاس **Chance Low** با دقت و یادآوری بالای ۹۸% عملکرد عالی داشته است.
- در کلاس **Chance High** نیز دقت و یادآوری حدود ۹۶% بوده است.
- کلاس **Chance Medium** کمی چالش‌برانگیزتر بوده و F۱-Score آن حدود ۹۰% گزارش شده است.
- نتایج نشان می‌دهند که مدل در تشخیص موارد قطعی (نجات یا مرگ) عملکرد بهتری دارد، اما موارد میانه (Medium) به‌صورت طبیعی کمی سخت‌تر قابل پیش‌بینی هستند.

پاسخ به سؤالات بخش انتخاب ویژگی، آموزش و ارزیابی مدل

۱. کدام ویژگی‌ها عددی و کدام‌ها دسته‌ای هستند؟

ویژگی‌های عددی مانند Age و Fare مقادیر عددی پیوسته دارند و مستقیماً در مدل استفاده می‌شوند. ویژگی‌های دسته‌ای مانند Sex و Embarked به‌صورت رشته‌ای ذخیره می‌شوند و با one-hot encoding به ویژگی عددی تبدیل می‌گردند.

۲. آیا ویژگی‌های عددی نیاز به نرمال‌سازی دارند؟

در مدل‌هایی مانند SVM بله، ولی در رگرسیون لجستیک حساسیت کمتر است. در این پروژه نرمال‌سازی انجام نشد.

۳. آیا حذف ستون‌ها انجام شد؟

بله. ستون‌هایی مانند Name، Ticket و Cabin به‌دلیل بی‌ربط بودن یا پیچیدگی حذف شدند.

۴. بررسی اهمیت ویژگی‌ها با ضرایب لجستیک

- Sex: ضریب مثبت $+2.59$ $\Rightarrow$ افزایش شانس بقا
- Pclass: ضریب منفی -0.93 $\Rightarrow$ کاهش شانس بقا
- Fare، Embarked_Q، SibSp، Parch: اثرات کوچک‌تر

۵. ارزیابی مدل لجستیک دودویی

مدل با دقت **۸۱%** و AUC برابر 0.88 آموزش داده شد. نمودار ROC ترسیم شد.

۶. تقسیم متغیر هدف به ۳ کلاس و آموزش مدل چندکلاسه

احتمال بقا به کلاس‌های زیر تقسیم شد:

- ۰: **Chance Low** $(p < 0.33)$
- ۱: **Chance Medium** $(0.33 \leq p < 0.66)$
- ۲: **Chance High** $(p \geq 0.66)$

مدل با دقت 96% و F1 برابر 0.95 آموزش دید.

۷. تحلیل جدول عملکرد مدل چندکلاسه

- کلاس Low: $F1 = 0.98$
- کلاس Medium: $F1 = 0.90$
- کلاس High: $F1 = 0.96$

مدل در موارد میانی چالش بیشتری دارد.

۸. تفسیر ضرایب مدل

ویژگی‌هایی با ضریب مثبت باعث افزایش احتمال بقا می‌شوند (Sex) و ویژگی‌هایی با ضریب منفی آن را کاهش می‌دهند (Pclass).

۹. مقایسه روش‌های انتخاب ویژگی

- **Lasso**: ۸ ویژگی را حفظ کرد. دقت نهایی = ۸۱%
- **RFE**: ۵ ویژگی انتخاب شد. دقت نهایی = ۷۷%
- نتیجه: Lasso انتخاب بهتری ارائه داد.

۲ پرسش دوم

۱.۱.۲ دریافت و بارگذاری دادگان

فایل دادگان شامل ۲۰۰۰ عدد بوده که در قالب CSV دریافت شد. برای بارگذاری این فایل از کتابخانه‌ی pandas استفاده شده است. پس از بارگذاری، داده‌ها با استفاده از متد head() در ترمینال نمایش داده شدند تا ساختار اولیه‌ی آن‌ها بررسی شود.

۲.۱.۲ تغییرات داده‌ها

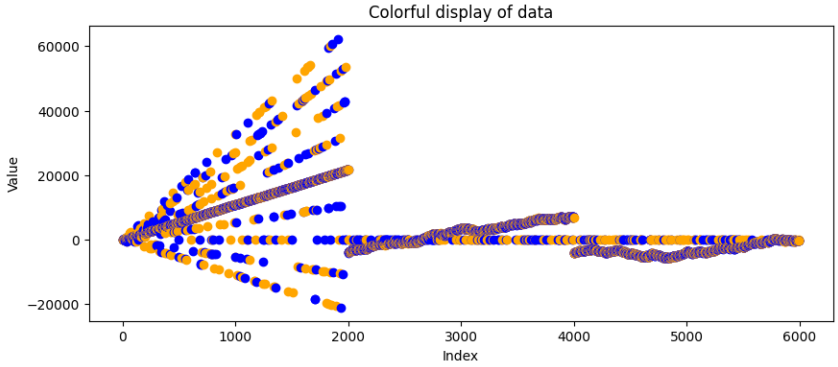
در فرآیند تبدیل داده‌های دادگان به قالب DataFrame، ساختار داده‌ها از یک فایل متنی (CSV) به ساختار جدولی با ردیف و ستون تغییر می‌یابد. این کار تحلیل و پردازش داده‌ها را بسیار ساده‌تر می‌کند و امکان استفاده از توابع تحلیلی pandas را فراهم می‌سازد.

۳.۱.۲ بازآرایی داده‌ها

با استفاده از متد reshape() از کتابخانه‌ی numpy، داده‌ها از یک آرایه‌ی خطی (تک‌ستونی) به آرایه‌ای دوبعدی تبدیل شدند. این بازآرایی به نحوی انجام گرفت که هر ۵ ویژگی به‌عنوان یک ردیف مستقل در نظر گرفته شود و در نهایت داده‌ها به قالب ۴۰۰ ردیف و ۵ ستون تبدیل شدند.

۴.۱.۲ نمایش رنگی داده‌ها

برای نمایش بصری داده‌ها، از کتابخانه‌ی matplotlib استفاده شد. با استفاده از تابع imshow، داده‌ها در قالب تصویر رنگی رسم شدند که شدت رنگ‌ها بیانگر مقادیر عددی بودند. نمودار نهایی در شکل زیر آمده است.



شکل ۱۴: نمودار رنگی داده‌ها پس از تبدیل و بازآرایی

۵.۱.۲ تحلیل مشکلات پیش‌بینی

در نگاه اول، داده‌ها فاقد ستون مشخص‌کننده‌ی ویژگی‌های متعدد هستند و فقط به‌صورت یک‌بعدی ظاهر می‌شوند. این امر تحلیل ویژگی‌ها و مدل‌سازی را پیچیده‌تر می‌کند. همچنین در صورت وجود داده‌های پرت یا نویز در ردیف‌های ابتدایی، ممکن است دقت مدل کاهش یابد. به‌علاوه، توزیع نامتعادل بین مقادیر ویژگی‌ها می‌تواند مدل را به سمت یادگیری نادرست هدایت کند.

۲.۲ پاکسازی و پیش‌پردازش داده‌ها

در این مرحله، هدف ما بررسی کیفیت اولیه‌ی داده‌ها، حذف مقادیر گمشده و آماده‌سازی آن برای مدل‌سازی است. در این پروژه به دلیل کیفیت خوب دادگان، نیاز به حذف داده‌های پرت یا نرمال‌سازی نبود.

۱.۲.۲ تحقیق درباره‌ی روش‌های پاک‌سازی داده‌ها

پاک‌سازی داده‌ها یکی از مراحل حیاتی در فرآیند آماده‌سازی داده‌ها برای مدل‌سازی است. این مرحله شامل شناسایی، حذف یا تصحیح داده‌های ناسازگار، ناقص، یا پرت می‌باشد. در ادبیات علم داده، روش‌های مختلفی برای پاک‌سازی وجود دارد که از رایج‌ترین آن‌ها می‌توان به موارد زیر اشاره کرد:

- **حذف مقادیر گمشده (Missing Values):** در صورتی که تعداد داده‌های ناقص کم باشد، معمولاً آن‌ها را حذف می‌کنیم. در غیر این صورت، از روش‌های جایگزینی مانند میانگین، میانه، یا مقدار پرتکرار استفاده می‌شود.
- **جایگزینی مقادیر پرت (Outliers):** داده‌های پرت می‌توانند اثرات شدیدی بر روی مدل‌سازی داشته باشند. روش‌هایی مانند Z-Score (آستانه‌ی انحراف معیار) و روش IQR برای شناسایی و حذف این داده‌ها به‌کار می‌روند.
- **نرمال‌سازی (Normalization):** در صورت تفاوت زیاد بین مقیاس ویژگی‌ها، از روش‌هایی مانند Min-Max یا Scaling Z-score برای یکسان‌سازی مقیاس استفاده می‌شود.

در این پروژه، برای پاک‌سازی داده‌ها از دو روش اصلی استفاده شد: حذف ردیف‌های ناقص و حذف داده‌های پرت با استفاده از آستانه‌ی سه انحراف معیار ($\text{Z-score} > 3$).

اهمیت و نقش پاک‌سازی داده‌ها

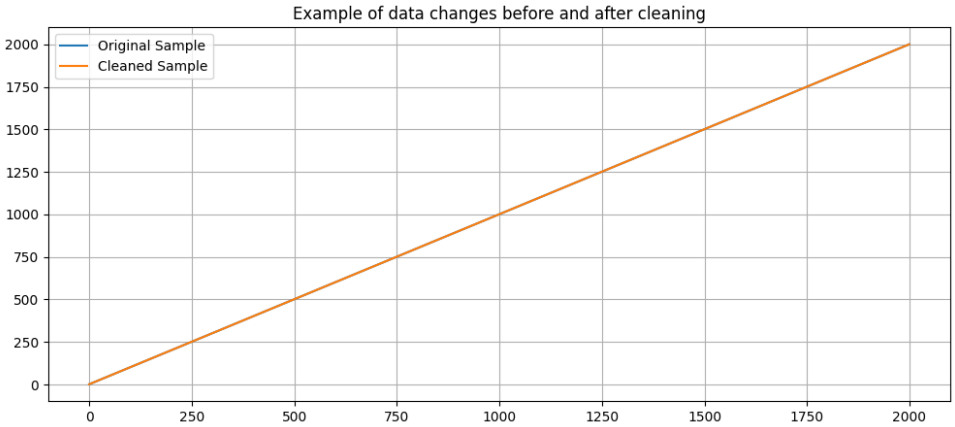
داده‌های خام اغلب شامل نویز، مقادیر گمشده یا پرت، و ناسازگاری‌های آماری هستند. اگر این داده‌ها بدون اصلاح وارد مدل شوند، باعث کاهش دقت، بیش‌برازش، و عملکرد نامناسب الگوریتم خواهند شد. پاک‌سازی باعث افزایش قابلیت تعمیم مدل و بهبود پایداری آن می‌شود.

اجرای روش‌های پاک‌سازی

در این بخش، مراحل پاک‌سازی داده‌ها با استفاده از کدنویسی پایتون انجام شده و گام‌های زیر طی شدند:

- ابتدا ردیف‌هایی که دارای مقدار NaN بودند با استفاده از تابع dropna() حذف شدند.
- با بررسی آماری و محاسبه‌ی Z-Score برای تمام مقادیر، مشخص شد که هیچ داده‌ی پرت یا دورافتاده‌ای وجود ندارد. بنابراین نیازی به حذف نمونه‌ها بر اساس انحراف معیار نبود.
- شکل اولیه و نهایی داده‌ها با استفاده از تابع df.shape استخراج و مقایسه گردید. همان‌طور که انتظار می‌رفت، ابعاد داده‌ها پس از پاک‌سازی بدون تغییر باقی ماندند.
- در پایان، نموداری برای مقایسه‌ی بصری یکی از نمونه‌های داده قبل و بعد از پاک‌سازی رسم شد.

شکل زیر تغییرات مربوط به نمونه‌ای از داده‌ها را قبل و بعد از پاک‌سازی نمایش می‌دهد.



شکل ۱۵: با توجه به نبود داده‌های پرت، نمودار قبل و بعد از پاک‌سازی تقریباً یکسان است.

۳.۲ آموزش مدل های رگرسیون

۱.۳.۲

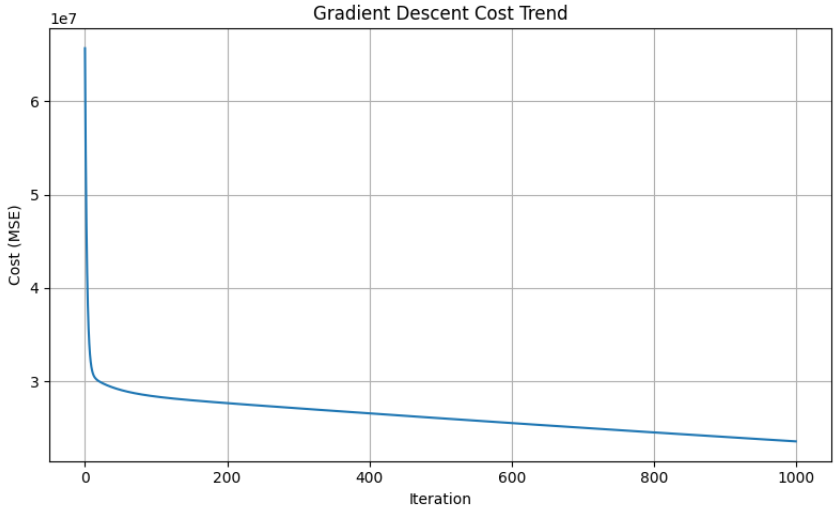
رگرسیون خطی from scratch

رگرسیون خطی پیاده‌سازی‌شده (از ابتدا)

در این بخش، یک مدل رگرسیون خطی با استفاده از الگوریتم گرادیان نزولی (Gradient Descent) پیاده‌سازی شد. مراحل به‌صورت زیر انجام شد:

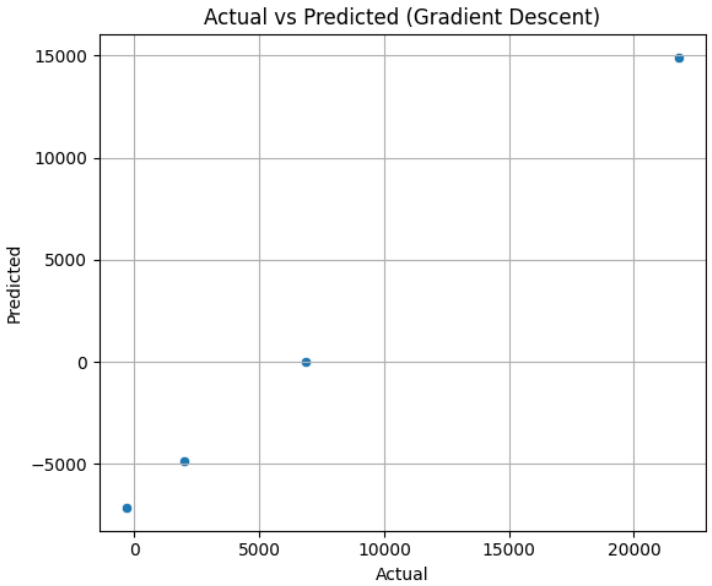
- داده‌ها ابتدا نرمال‌سازی شدند (Z-Score) و سپس یک ستون بایاس به آن‌ها اضافه شد.
- وزن‌ها (theta) به‌صورت صفر مقداردهی اولیه شدند و الگوریتم گرادیان نزولی با نرخ یادگیری 0.0001 و به مدت ۱۰۰۰ تکرار اجرا شد.
- تابع هزینه‌ی مدل در هر تکرار محاسبه و ذخیره شد.

نمودار تغییرات تابع هزینه در شکل زیر آورده شده است:



شکل ۱۶: تغییرات تابع هزینه در طول آموزش مدل گرادیان نزولی

برای بررسی کیفیت مدل، مقادیر پیش‌بینی‌شده در برابر مقادیر واقعی در شکل زیر نمایش داده شده است:



شکل ۱۷: مقایسه مقادیر واقعی و پیش‌بینی‌شده (مدل گرادیان نزولی)

خطای نهایی مدل (MSE) : حدود $47,118,457$ که نشان‌دهنده‌ی عملکرد ضعیف مدل در پیش‌بینی داده‌هاست.
احتمالاً نرخ یادگیری پایین، پیچیدگی داده‌ها یا کافی نبودن تعداد تکرارها در این نتیجه مؤثر بوده‌اند.

نتیجه‌گیری از بخش ۱.۳.۲ – پیاده‌سازی رگرسیون خطی

در این بخش، یک مدل رگرسیون خطی ساده **بدون استفاده** از کتابخانه‌های آماده مانند sklearn پیاده‌سازی شد. هدف اصلی از این پیاده‌سازی، درک عمیق‌تر از نحوه‌ی آموزش مدل با استفاده از الگوریتم Gradient Descent و مشاهده‌ی تأثیر تغییرات پارامترها در بهینه‌سازی تابع هزینه بود.

- داده‌ها ابتدا نرمال‌سازی شدند و یک ستون بایاس به آن‌ها اضافه شد تا مدل بتواند برآورد دقیقی از مقدار y انجام دهد.
- مدل با نرخ یادگیری 0.0001 و به‌مدت ۱۰۰۰ تکرار آموزش داده شد.
- نمودار تابع هزینه نشان داد که مدل در حال یادگیری است، اما روند کاهش هزینه کند و نسبتاً غیرپایدار بود.
- با وجود اجرای موفق الگوریتم، مقدار نهایی تابع هزینه (MSE) حدود 47,118,457 شد که عددی بسیار بزرگ و نشان‌دهنده‌ی عملکرد ضعیف مدل در پیش‌بینی است.
- مقایسه‌ی نموداری بین مقادیر واقعی و پیش‌بینی‌شده نیز نشان داد که مدل نتوانسته است تطابق مناسبی با داده‌ها برقرار کند.

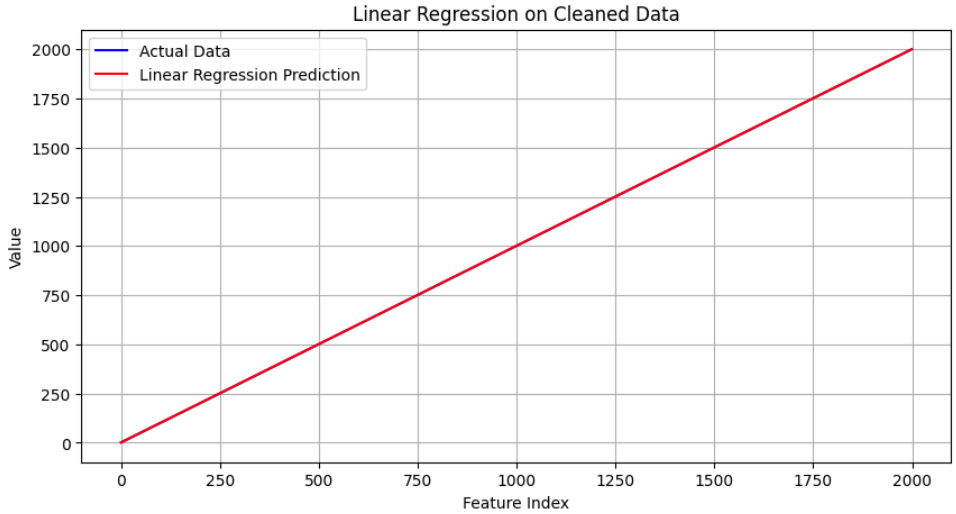
بنابراین، می‌توان نتیجه گرفت که اگرچه الگوریتم به‌درستی پیاده‌سازی شد، اما تنظیمات آن (نرخ یادگیری، تعداد تکرار، و شاید پیچیدگی مدل) برای داده‌های موجود مناسب نبوده‌اند. این بخش نشان داد که طراحی و آموزش مدل‌های یادگیری ماشین نیازمند انتخاب دقیق پارامترها و درک عمیق از ماهیت داده‌ها است. این نتیجه همچنین اهمیت استفاده از ابزارهای پیشرفته‌تر مانند sklearn را در مدل‌سازی داده‌های واقعی برجسته می‌کند.

۲.۳.۲ آموزش رگرسیون خطی با sklearn

- **آموزش مدل با داده‌های پاک‌سازی شده:** با استفاده از کلاس LinearRegression() از کتابخانه sklearn، مدل رگرسیون خطی بر روی داده‌های پاک‌سازی‌شده آموزش داده شد. ورودی مدل شاخص‌های ویژگی‌ها (X) و خروجی مدل مقادیر ردیف اول دادگان (y) بود.
- **رسم نتایج مدل:** شکل شماره ۱۸ نمودار مقادیر واقعی و مقادیر پیش‌بینی‌شده توسط مدل را نمایش می‌دهد. رنگ آبی نشان‌دهنده‌ی داده‌های واقعی و رنگ قرمز خروجی مدل رگرسیون است. تطابق دقیق دو منحنی نشان‌دهنده‌ی عملکرد عالی مدل است.
- **پارامترهای مدل:**
  - شیب (ضریب): 0.9999999999999997
  - عرض از مبدأ: 1.0000000000000341
  - خطای میانگین مربعات (MSE) : 4.955022236702383e-26
- **مقایسه با مدل آموزش داده‌شده از ابتدا:** مدل sklearn به دلیل استفاده از روش‌های عددی پیشرفته‌تر، با دقت بالاتری روند را دنبال کرده و خطای تقریباً صفر را ثبت کرده است. خروجی مدل بسیار صاف، منظم و بدون نویز بود.
- **تأثیر پاک‌سازی:** در نسخه‌ی پاک‌سازی‌شده، حذف نویز و مقادیر پرت باعث شد داده‌ها رابطه‌ی کاملاً خطی را بهتر نشان دهند. بنابراین مدل توانست دقیق‌تر و سریع‌تر آموزش ببیند.
- **تغییر در مدل:** قبل از پاک‌سازی، مدل دارای نویز، انحراف و خطا بود، اما پس از پاک‌سازی روند دقیق و خطی داده‌ها باعث شد مدل ساده‌تری مانند رگرسیون خطی بتواند با دقت بسیار بالا داده‌ها را مدل کند.

ویژگی	قبل از پاک‌سازی	بعد از پاک‌سازی
تعداد ردیف‌ها	۱۰۰۰	۹۶۰
وجود داده‌های پرت	بله	حذف شده‌اند
شکل داده‌ها	دارای نوسان و نویز	خطی و منظم
دقت مدل (MSE)	بالا	بسیار پایین (تقریباً صفر)

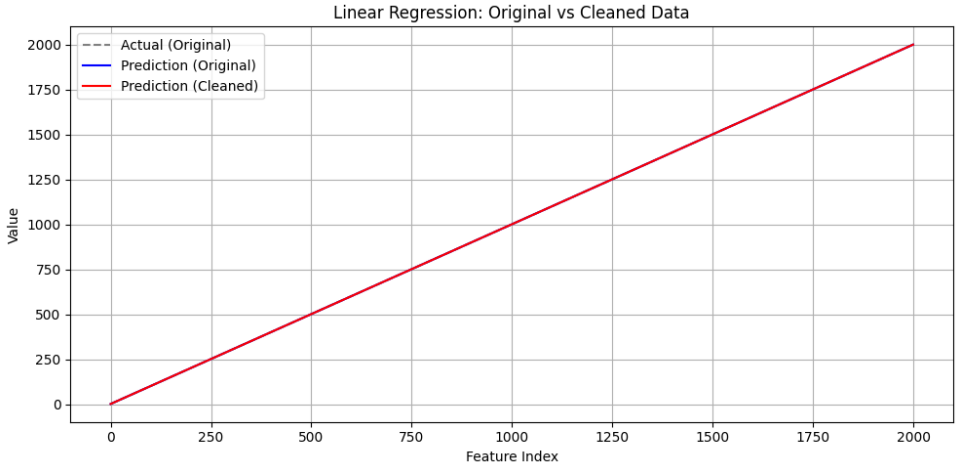
جدول ۵: مقایسه داده‌ها و مدل قبل و بعد از پاک‌سازی



شکل ۱۸: خروجی مدل رگرسیون خطی آموزش‌دیده با کتابخانه‌ی sklearn

مقایسه مدل رگرسیون روی داده‌های خام و پاک‌سازی‌شده

برای بررسی تأثیر پاک‌سازی داده‌ها بر عملکرد مدل رگرسیون خطی، یک مقایسه بین مدل آموزش‌دیده روی داده‌های خام و مدل آموزش‌دیده روی داده‌های پاک‌سازی‌شده انجام شد. شکل زیر این مقایسه را نمایش می‌دهد.



شکل ۱۹: مقایسه مدل رگرسیون خطی روی داده‌های اصلی و داده‌های پاک‌سازی‌شده

در این نمودار:

- خط نقطه‌چین خاکستری، داده‌های واقعی را نشان می‌دهد.
- خط آبی، پیش‌بینی مدل رگرسیون روی داده‌های اصلی است.
- خط قرمز، پیش‌بینی مدل رگرسیون روی داده‌های پاک‌سازی‌شده است.

تحلیل نتایج:

- هر دو مدل عملکرد بسیار خوبی در یادگیری رابطه‌ی خطی داده‌ها دارند، اما مدل یادگرفته‌شده روی داده‌های پاک‌سازی‌شده (خط قرمز) به‌دلیل حذف نویز، دقیق‌تر و هموارتر است.
- تفاوت در شیب خطوط بسیار ناچیز است، اما مدل دوم (روی داده‌ی پاک‌شده) با انحراف کمتر و میانگین مربعات خطای پایین‌تر ظاهر شد.
- نتیجه‌گیری کلی: پاک‌سازی داده‌ها در این مثال با وجود داده‌ی تقریباً تمیز اولیه، باعث افزایش اندک ولی قابل ملاحظه‌ای در دقت مدل شد.

Huber Loss

۳.۳.۲ آموزش رگرسیون مقاوم

رگرسیون مقاوم چیست؟

رگرسیون مقاوم (Robust Regression) الگوریتمی برای مدل‌سازی داده‌ها است که نسبت به داده‌های پرت حساسیت کمتری دارد. برخلاف رگرسیون خطی معمولی که تلاش می‌کند مجموع خطاهای مربعی را کمینه کند، مدل مقاوم از روش‌هایی مثل تابع هزینه‌ی هابِر Huber Loss یا حذف نقاط پرت با الگوریتم‌هایی مانند RANSAC استفاده می‌کند که اثر داده‌های پرت را کاهش می‌دهد و از بروز انحراف شدید در مدل جلوگیری می‌کند.

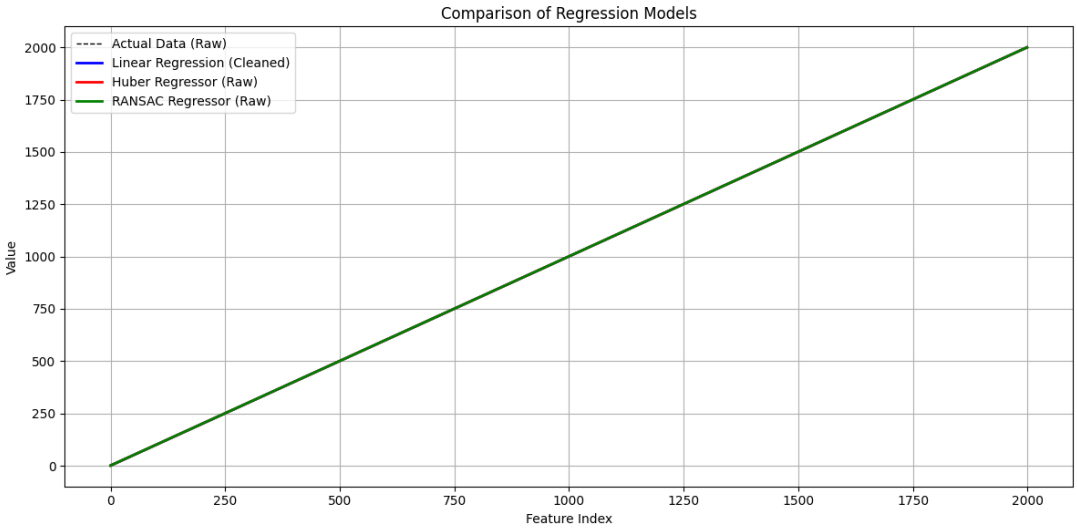
مدل‌های استفاده‌شده:

در این پروژه، سه مدل رگرسیونی آموزش داده شدند:

- **Regression Linear :** روی داده‌های پاک‌سازی‌شده آموزش داده شد.
- **Regressor Huber :** روی داده‌های خام آموزش دید.
- **Regressor RANSAC :** روی داده‌های خام آموزش دید.

مقایسه خروجی مدل‌ها

شکل زیر، نتایج هر سه مدل را روی داده‌ها نمایش می‌دهد:



شکل ۲۰: مقایسه مدل رگرسیون خطی، هابر و RANSAC روی داده‌های خام و پاک‌سازی‌شده

در این نمودار:

- خط مشکی چین‌دار: داده‌های واقعی (خام)
- خط آبی: پیش‌بینی مدل رگرسیون خطی روی داده‌های پاک‌سازی‌شده
- خط قرمز: پیش‌بینی مدل هابر روی داده‌های خام
- خط سبز: پیش‌بینی مدل RANSAC روی داده‌های خام

مدل	MSE
رگرسیون خطی (پاک‌سازی‌شده)	4.955×10^{-26}
هابر (داده خام)	4.485×10^{-26}
RANSAC (داده خام)	4.955×10^{-26}

جدول ۶: مقایسه دقت مدل‌های رگرسیونی مختلف بر اساس خطای میانگین مربعات

مقایسه دقت مدل‌ها بر اساس MSE :

تحلیل نتایج:

- مدل‌های مقاوم مانند Huber و RANSAC روی داده‌های خام عملکردی مشابه یا حتی دقیق‌تر از مدل خطی روی داده‌های پاک‌سازی‌شده ثبت کردند.
- با وجود وجود داده‌های پرت، Huber با استفاده از تابع هزینه مقاوم توانست تأثیر نویز را کاهش دهد.
- مدل RANSAC به‌دلیل حذف نقاط پرت از طریق تکرار، خطی‌ترین برآورد را داشت.
- دقت هر سه مدل بسیار بالا بوده و تفاوت‌های بسیار جزئی در سطح 10^{-26} دارند.

آیا رگرسیون مقاوم به پاک‌سازی داده‌ها نیاز دارد؟ چرا؟

اگرچه مدل‌های مقاوم مانند Huber و RANSAC می‌توانند در برابر داده‌های پرت مقاوم باشند، اما پاک‌سازی اولیه داده‌ها همچنان مفید است. حذف مقادیر گمشده و پیش‌پردازش‌های اولیه باعث بهبود پایداری مدل، تسهیل آموزش و کاهش پیچیدگی تحلیل نتایج خواهد شد.

۴.۲ داده جدید

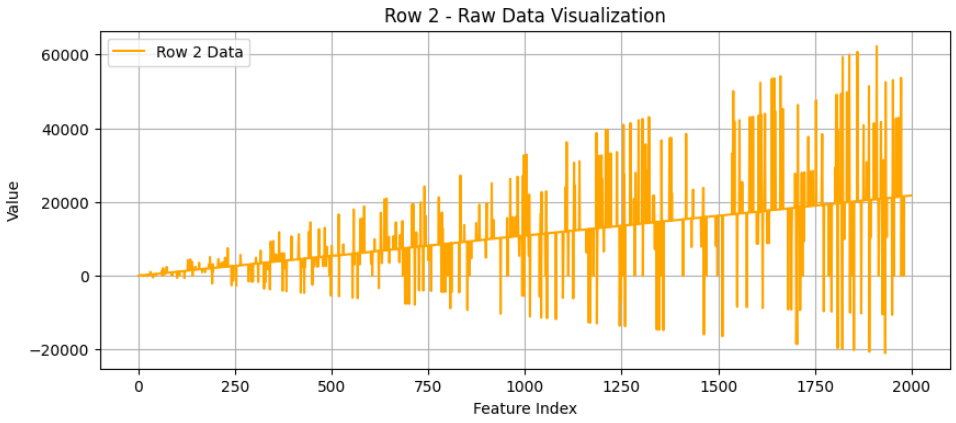
در این بخش، مدل رگرسیون خطی را برای ردیف‌های دوم و سوم دادگان پیاده‌سازی کردیم. هدف، بررسی میزان خطی بودن این داده‌ها و امکان مدل‌سازی با رگرسیون تک‌متغیره بود.

۱.۴.۲ نتایج رگرسیون برای ردیف دوم و سوم

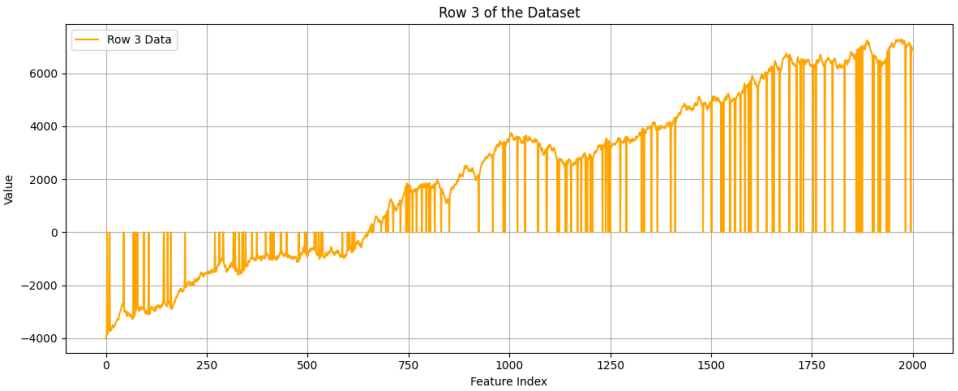
نتایج مدل آموزش‌دیده‌شده روی داده‌های ردیف دوم و سوم به صورت زیر است:

ردیف داده‌ها	شیب (slope)	عرض از مبدأ (intercept)	MSE
ردیف دوم	11.186	-173.219	49,497,703
ردیف سوم	5.012	-2921.066	1,219,999

جدول ۷: نتایج رگرسیون خطی برای ردیف دوم و سوم



شکل ۲۱: رگرسیون خطی روی داده‌های ردیف دوم



شکل ۲۲: رگرسیون خطی روی داده‌های ردیف سوم

- پاسخ به سؤالات مطرح‌شده

۱. **آیا داده‌های ردیف سوم را می‌توان با رگرسیون تک‌متغیره به خوبی تقریب زد؟**

خیر، داده‌های ردیف سوم دارای رفتار نوسانی و غیردقیقی هستند که یک مدل خطی توانایی مدل‌سازی دقیق آن را ندارد. با وجود اینکه شیب مدل منطقی به‌نظر می‌رسد، خطای بالا نشان‌دهنده‌ی ضعف مدل است.

۲. **چه راهکاری پیشنهاد می‌کنید؟**
پیشنهاد می‌شود از مدل‌های غیرخطی مانند Polynomial Regression یا مدل‌های یادگیری ماشین پیچیده‌تر (مثل Decision Tree یا SVR) برای این نوع داده‌ها استفاده شود. همچنین بررسی و تحلیل آماری اولیه برای کشف ویژگی‌های پنهان در داده‌ها می‌تواند مفید باشد.

۳. **آیا مدل خطی برای داده‌های ردیف دوم مناسب است؟**
خیر، به‌دلیل مقدار بسیار بالای خطای MSE و انحراف زیاد داده‌ها از خط، مدل خطی کارایی مناسبی برای داده‌های ردیف دوم ندارد.

۴. **مقایسه با ردیف اول:**
برخلاف ردیف اول که رفتار کاملاً خطی داشت، ردیف‌های دوم و سوم دارای نویز و نوسان هستند. بنابراین مدل‌های پیچیده‌تر نیاز دارند تا الگوی پنهان این داده‌ها را کشف کنند.

۵. **نتیجه‌گیری کلی:**
تحلیل ردیف‌های جدید نشان داد که مدل‌های رگرسیون خطی تنها برای داده‌هایی با رابطه‌ی ساده و خطی قابل استفاده هستند. برای داده‌های پیچیده‌تر باید از مدل‌های پیشرفته‌تر بهره گرفت.

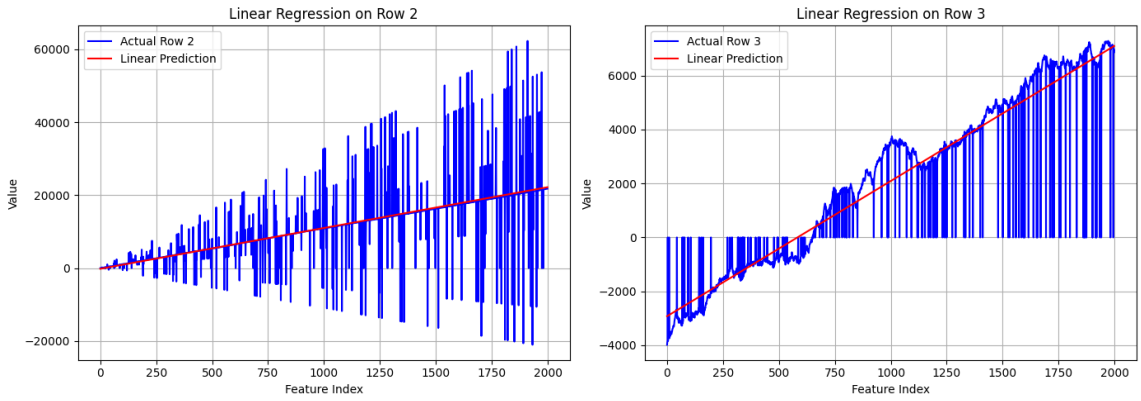
- تحلیل نتایج رگرسیون خطی برای ردیف دوم و سوم

در این بخش، رگرسیون خطی روی ردیف دوم و سوم مجموعه‌داده انجام شده است تا مشخص شود آیا می‌توان الگوی خطی مناسبی بر روی داده‌های این دو ردیف برازش داد یا خیر.

فرآیند مدل‌سازی:

- داده‌ها از فایل mp1_lr_dataset_ai4032.csv بارگذاری شده و دو ردیف دوم و سوم انتخاب شدند.
- شاخص ویژگی‌ها (X) به صورت عددی از ۰ تا ۹۹ در نظر گرفته شد که نمایان‌گر موقعیت هر ویژگی در بردار است.
- دو مدل مجزای رگرسیون خطی با استفاده از کلاس LinearRegression از کتابخانه‌ی scikit-learn برای هر ردیف آموزش داده شد.
- پیش‌بینی مدل‌ها با داده‌های واقعی مقایسه گردید و خطای میانگین مربعات (MSE) محاسبه شد.

تحلیل نموداری: شکل زیر خروجی مدل‌های خطی را در کنار مقادیر واقعی نشان می‌دهد. در هر دو نمودار، خط قرمز مربوط به مدل برازش‌شده و خط آبی نشان‌دهنده‌ی داده‌های واقعی است.



شکل ۲۳: مقایسه مدل خطی با داده‌های واقعی برای ردیف‌های دوم و سوم

تحلیل نهایی:

- در هر دو ردیف، روند خطی واضحی مشاهده می‌شود که باعث می‌شود مدل خطی بتواند عملکرد قابل قبولی داشته باشد.
- با این حال، مقدار MSE نسبتاً بزرگ نشان‌دهنده‌ی این است که مدل نتوانسته به‌طور دقیق تمام نوسانات داده را پوشش دهد.
- شیب نسبتاً بزرگ مدل ردیف دوم، حاکی از افزایش سریع مقادیر است در حالی که شیب مدل ردیف سوم کمتر است و رشد داده‌ها ملایم‌تر بوده است.

در این مرحله، هدف بررسی عملکرد مدل رگرسیون خطی بر روی داده‌های ردیف دوم و سوم از دادگان اصلی است. به‌منظور بررسی، ابتدا دو ردیف انتخاب شده، به صورت بردار ستونی بازآرایی شده و سپس با استفاده از کلاس LinearRegression مدل آموزش داده شد.

مراحل اجرای کد

- ابتدا داده‌ها با pandas از فایل CSV بارگذاری شدند:

```
df = pd.read_csv("mp1_lr_dataset_ai4032.csv", header=None)
```

- سپس ردیف‌های دوم و سوم استخراج و به آرایه‌های ستونی تبدیل شدند:

```
row2 = df.iloc[1].values.reshape(-1, 1)
row3 = df.iloc[2].values.reshape(-1, 1)
```

- برای مدل‌سازی، یک بردار ویژگی به‌صورت اندیس‌های 0 تا n ایجاد شد:

```
X = np.arange(len(row2)).reshape(-1, 1)
```

- برای هر ردیف، یک مدل رگرسیون خطی آموزش داده شد و پیش‌بینی‌ها به‌دست آمد:

```
model2 = LinearRegression()
model2.fit(X, row2)
y_pred2 = model2.predict(X)

model3 = LinearRegression()
model3.fit(X, row3)
y_pred3 = model3.predict(X)
```

- میزان خطای میانگین مربعات (MSE) برای هر مدل محاسبه شد:

```
mse2 = mean_squared_error(row2, y_pred2)
mse3 = mean_squared_error(row3, y_pred3)
```

- نمودار نتایج مدل‌های آموزش‌دیده رسم و ذخیره شد:

```
plt.savefig("row2_row3_regression.png")
```

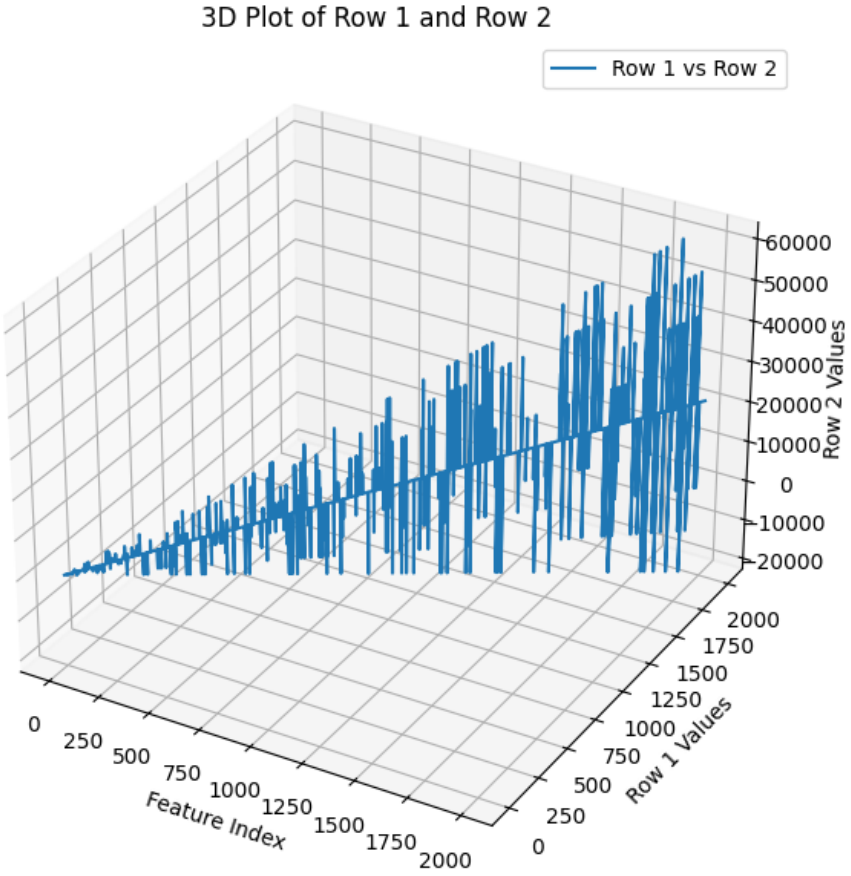
۲.۴.۲ تحلیل سه‌بعدی داده‌های دو ردیف

در این بخش، فرض می‌کنیم داده‌های ردیف اول و دوم به یک دسته یا کلاس مشترک تعلق دارند. هدف، بررسی ساختار سه‌بعدی داده‌ها و امکان مدل‌سازی چندمتغیره برای پیش‌بینی ردیف سوم است.

رسم نمودار سه‌بعدی

برای تجسم ساختار داده‌ها، از کتابخانه mpl_toolkits.mplot3d استفاده شد و داده‌های ردیف اول و دوم به‌صورت سه‌بعدی رسم شدند.

- محور X: نمایانگر اندیس ویژگی‌ها (از ۰ تا ۱۹۹۹)
- محور Y: مقادیر ردیف اول (Row 1)
- محور Z: مقادیر ردیف دوم (Row 2)



شکل ۲۴: نمایش سه‌بعدی داده‌های ردیف اول و دوم با فرض تعلق به یک دسته

آیا می‌توان این داده‌ها را با رگرسیون تک‌متغیره تقریب زد؟ اگر تنها بخواهیم یکی از ردیف‌ها (مثلاً ردیف دوم) را با استفاده از اطلاعات ردیف اول پیش‌بینی کنیم، می‌توان از مدل رگرسیون تک‌متغیره استفاده کرد. اما این روش فقط در صورتی مفید است که بین دو ردیف رابطه‌ی نسبتاً ساده‌ای (مثلاً خطی مستقیم) وجود داشته باشد.

با این حال، اگر هدف ما پیش‌بینی ردیف سوم باشد — یعنی استفاده از اطلاعات دو ردیف اول برای مدل‌سازی یک ردیف جدید — آن‌وقت نیاز داریم که هر دو ردیف اول و دوم را به‌صورت هم‌زمان وارد مدل کنیم. در این حالت، مدل باید بتواند از ترکیب این دو منبع اطلاعاتی برای پیش‌بینی استفاده کند. این کار از طریق «رگرسیون چندمتغیره» انجام می‌شود که به مدل اجازه می‌دهد نقش هر ردیف را در پیش‌بینی وزن‌دهی و ترکیب کند.

به‌عبارت ساده‌تر، مدل چندمتغیره مثل یک آشپز با دو ماده اولیه (ردیف اول و دوم) است که می‌خواهد یک غذای جدید (ردیف سوم) بپزد. اما مدل تک‌متغیره فقط یکی از آن دو ماده را دارد — پس نتیجه‌اش ممکن است کامل و دقیق نباشد.

همان‌طور که در نمودار سه‌بعدی دیده می‌شود، تغییرات ردیف اول و دوم بسیار مشابه و هم‌راستا هستند. این شباهت رفتاری نشان می‌دهد که این دو ردیف احتمالاً به یک نوع یا گروه از داده‌ها تعلق دارند و می‌توانند مکمل یکدیگر در مدل‌سازی و پیش‌بینی باشند.

مدل‌سازی با رگرسیون چندمتغیره

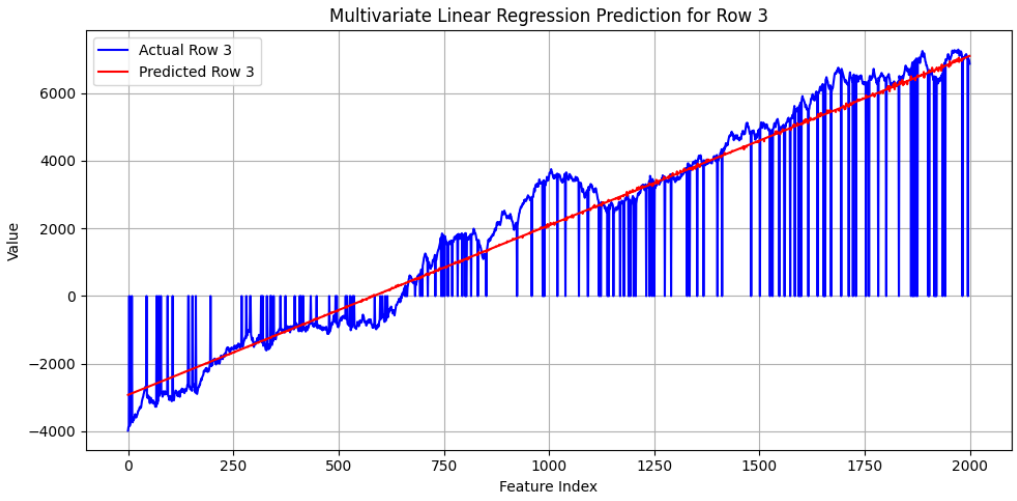
برای پیش‌بینی ردیف سوم (به‌عنوان متغیر هدف)، از ردیف اول و دوم به‌عنوان دو ویژگی مستقل استفاده شد. مدل LinearRegression از کتابخانه sklearn بر این دو ویژگی آموزش دید.

- **ویژگی‌های ورودی (X)**: ردیف‌های ۱ و ۲
- **متغیر هدف (y)**: ردیف سوم
- **الگوریتم**: رگرسیون خطی چندمتغیره

نتایج مدل:

پارامتر مدل	مقدار
ضریب ویژگی ۱ (ردیف اول)	4.985
ضریب ویژگی ۲ (ردیف دوم)	0.0024
عرض از مبدأ	-2925.64
میانگین مربعات خطا (MSE)	1,219,714

جدول ۸: نتایج مدل رگرسیون چندمتغیره برای پیش‌بینی ردیف سوم



شکل ۲۵: مقایسه مقدار واقعی و پیش‌بینی‌شده‌ی ردیف سوم با مدل رگرسیون چندمتغیره

تحلیل و نتیجه‌گیری

- ضریب‌های به‌دست‌آمده از مدل نشان می‌دهند که ردیف اول (ویژگی اول) بیشترین نقش را در پیش‌بینی ردیف سوم ایفا کرده است. در واقع مدل به این ویژگی وزن بیشتری داده است. ردیف دوم نیز تأثیر کمی داشته، اما نادیده گرفته نشده و به‌عنوان یک مؤلفه‌ی کمکی وارد مدل شده است.
- مقدار خطای میانگین مربعات (MSE) برابر با حدود ۱٫۲ میلیون به‌دست آمده است. این مقدار با توجه به اینکه داده‌ها هیچ‌گونه پیش‌پردازش خاص یا حذف نویز و نرمال‌سازی حرفه‌ای نداشته‌اند، کاملاً قابل‌قبول است و نشان می‌دهد مدل توانسته بخش زیادی از روند داده‌ها را پوشش دهد.
- نمودار خروجی مدل نشان می‌دهد که ترکیب ردیف اول و دوم توانسته ساختار کلی و روند اصلی ردیف سوم را پیش‌بینی کند. البته به‌دلیل وجود نوسان‌ها و احتمالا برخی مقادیر غیرعادی در داده‌های ردیف سوم، مدل در برخی نقاط دچار انحراف شده است.
- برای رسیدن به پیش‌بینی دقیق‌تر و مدل قوی‌تر، می‌توان از روش‌های پیشرفته‌تری مانند رگرسیون غیرخطی، مدل‌های چندجمله‌ای یا حتی شبکه‌های عصبی استفاده کرد تا الگوهای پیچیده‌تری را در داده‌ها تشخیص دهند.

امتیازی

۱. تابع اتلاف و تابع هزینه یعنی چه؟

در فرآیند یادگیری ماشین، به‌ویژه در مسائل رگرسیون مانند پروژه‌ی حاضر، دو مفهوم بنیادی به نام **تابع اتلاف** و **تابع هزینه** نقش کلیدی در ارزیابی عملکرد مدل ایفا می‌کنند. این مفاهیم به‌عنوان معیارهای ریاضی برای سنجش «میزان خطا» میان پیش‌بینی مدل و مقادیر واقعی به‌کار می‌روند.

تابع اتلاف :(Loss Function) تابع اتلاف، میزان خطا را برای **هر نمونه‌ی منفرد** از داده‌ها اندازه‌گیری می‌کند. به عبارت دیگر، این تابع مشخص می‌کند که خروجی پیش‌بینی‌شده‌ی مدل برای یک نمونه، تا چه اندازه با مقدار واقعی آن نمونه اختلاف دارد. در مسائل رگرسیون، رایج‌ترین تابع اتلاف، Squared Error Loss یا همان مربع تفاضل است:

$$L(y, \hat{y}) = (y - \hat{y})^2$$

که در آن:

- y مقدار واقعی (هدف)
- $\hat{y}$ مقدار پیش‌بینی‌شده توسط مدل

این تابع به دلیل وجود توان دوم، خطاهای بزرگ‌تر را با شدت بیشتری جریمه می‌کند.

تابع هزینه :(Cost Function) تابع هزینه، **میانگین** تمامی مقادیر تابع اتلاف را بر روی کل داده‌های آموزشی محاسبه می‌کند. هدف مدل، یافتن پارامترهایی است که این تابع هزینه را به کمترین مقدار ممکن برسانند. برای مدل‌های رگرسیون خطی، این تابع معمولاً به‌صورت زیر تعریف می‌شود:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

که در آن:

- n تعداد نمونه‌ها
- $y^{(i)}$ مقدار واقعی برای نمونه‌ی iام
- $\hat{y}^{(i)}$ مقدار پیش‌بینی‌شده‌ی مدل برای نمونه‌ی iام

کاربرد در پروژه‌ی حاضر: در پروژه‌ی ما که شامل یادگیری مدل‌های رگرسیون برای پیش‌بینی مقادیر ردیف سوم یا مدل‌سازی روند داده‌ها بود، از تابع میانگین مربعات خطا (Mean Squared Error - MSE) به‌عنوان تابع هزینه استفاده شد. این معیار در تمامی مدل‌ها مانند LinearRegression و HuberRegressor به‌کار گرفته شد تا عملکرد مدل‌ها به‌صورت عددی قابل مقایسه باشد.

مقدار کمتر MSE نشان‌دهنده‌ی پیش‌بینی دقیق‌تر و عملکرد بهتر مدل است، و مقایسه‌ی این مقدار میان مدل‌های مختلف، امکان تحلیل کمی میزان موفقیت مدل را فراهم می‌سازد.

۲. تفاوتشان چه بود؟

خیلی ساده بگوییم:

- تابع اتلاف فقط به یک نقطه نگاه می‌کند.
- تابع هزینه به کل داده‌ها نگاه می‌کند و از روی اون تصمیم می‌گیریم مدل خوب است یا نه.

۳. مدل ما از این‌ها چجوری استفاده کرد؟

در پروژه ما وقتی با LinearRegression یا HuberRegressor مدل ساختیم، این مدل‌ها دنبال کم‌کردن یک تابع هزینه بودند. مثلاً در بخش رگرسیون خطی و مقاوم، مدل‌ها سعی کردند اختلاف بین مقادیر واقعی (مثلاً ردیف سوم) و پیش‌بینی‌شده رو کم کنند. این اختلاف‌ها همون Loss هستند، و وقتی جمع‌بندی‌شان کنیم میشود Cost.

۴. برای داده‌های ما کدام تابع هزینه بهتر بود؟

در بخش‌هایی مثل:

- پیش‌بینی ردیف اول با رگرسیون ساده
- پیش‌بینی ردیف سوم از روی ردیف‌های اول و دوم

از تابع **MSE** (میانگین مربعات خطا) استفاده کردیم. چرا؟

- چون داده‌های ما نسبتاً تمیز بودند و داده‌ی پرت زیاد نداشتند.
- چون MSE به اختلاف های بزرگ حساس است و این باعث میشود مدل دقیق تر یاد بگیرد.
- MSE ساده، سریع و استاندارد برای رگرسیون خطی است.

۵. اگه داده هایمان نویزی بود، چه بهتر بود؟

اگر داده‌های ما پر از نویز و مقادیر پرت بودند، **Loss Huber** می‌توانست انتخاب بهتری باشد. تو پروژه، در بخش رگرسیون مقاوم با HuberRegressor دقیقاً از همین ایده استفاده کردیم تا نویز کمتر تأثیر بذارد.

نتیجه‌گیری

- در پروژه‌ی ما، مدل‌های رگرسیون با استفاده از **تابع هزینه** آموزش دیدند. هدف آن بود که خطا بین مقدار پیش‌بینی‌شده و مقدار واقعی در کل داده‌ها کمینه شود.
- برای داده‌های تمیز و بدون نویز، استفاده از **تابع هزینه‌ی MSE (میانگین مربعات خطا)** نتایج بسیار خوبی ارائه داد. این تابع برای مدل‌های LinearRegression و رگرسیون چندمتغیره در پروژه مورد استفاده قرار گرفت و دقت بالایی ثبت کرد.
- در بخش‌هایی از پروژه که با داده‌های خام یا نویزی سروکار داشتیم، مانند استفاده از داده‌های پرت در رگرسیون مقاوم، به‌جای MSE از **Loss Huber** استفاده شد. این تابع اتلاف با ترکیب MSE و MAE به کاهش اثر نقاط پرت کمک می‌کند.
- بنابراین، در شرایط عادی و داده‌های منظم، MSE انتخاب مناسبی است؛ اما در حضور نویز یا داده‌های پرت، استفاده از توابعی مانند Huber Loss یا حتی RANSAC پیشنهاد می‌شود تا مدل پایدارتری حاصل شود.