Ikra sarıçiçek 210408032

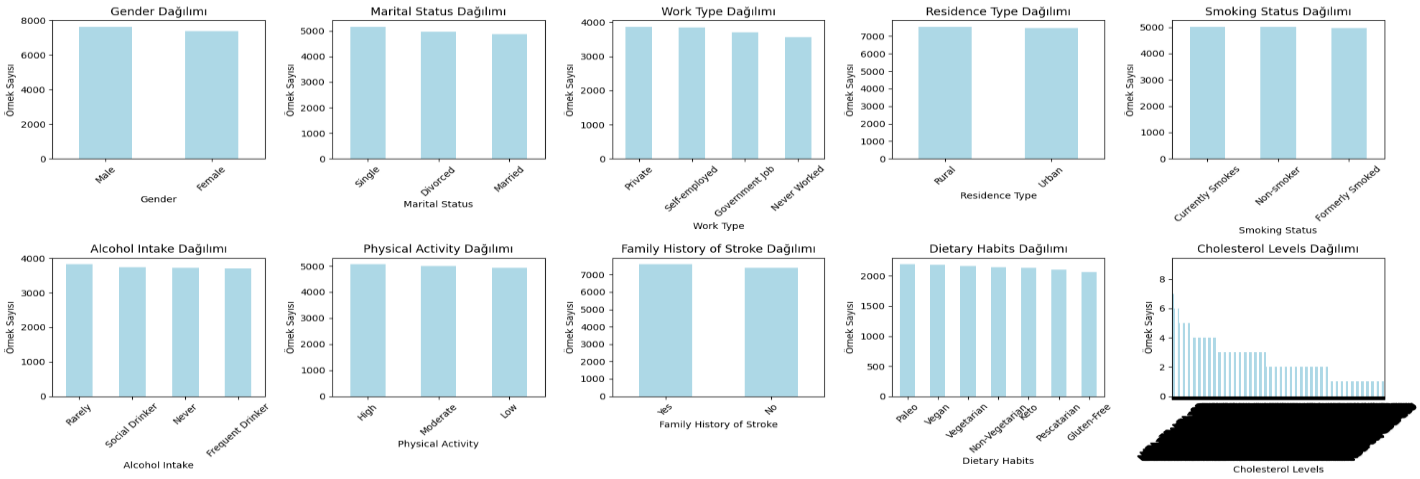
2024-2025 SPRING SEMESTER

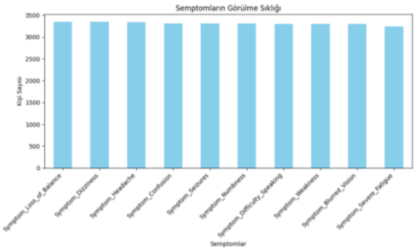
MACHINE LEARNING LECTURE TERM PROJECT

a) Dataset Overview

- **Dataset Name:** Stroke Prediction
- **Project Objective:**
  The goal of this project is to predict the risk of stroke in individuals by comparing various machine learning and deep learning algorithms to identify the model with the highest accuracy. This analysis aims to support early diagnosis and prevention efforts in the healthcare field.
- **Dataset Web Link:**https://www.kaggle.com/datasets/teamincribo/stroke-prediction
- **Number of Records:**15000
- **Number of Features:** 22
- **Number of Target Classes (if applicable):**Diagnosis sütunu(Stroke ,No Stroke)
- **Number of Examples per Class:**No Stroke 7532 ,Stroke 7468
- **Number of Missing (NULL) Values:** Symptoms: 2,500 missing values. All other columns: No missing values.
- **Which Features Are Non-Numeric?**Patient Name ,Gender, Marital Status,Work Type,Residence Type,Smoking Status,Alcohol Intake,Physical Activity,Family History of Stroke,Dietary Habits,Cholesterol Levels,Symptoms,Diagnosis

b) Categorical Feature Distribution



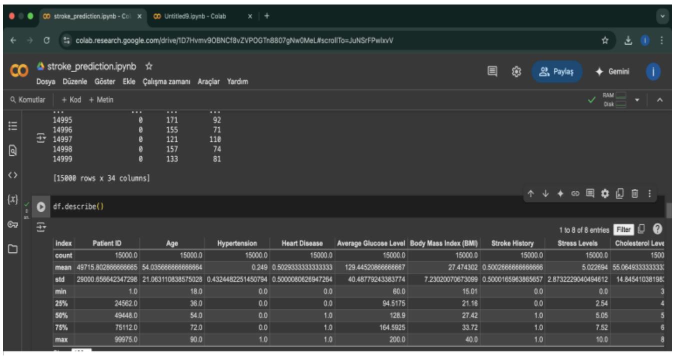


In the code, distributions for these columns were generated using commands such as value_counts() and plt.bar. Visualizations were created using **matplotlib** and **seaborn**. For each categorical feature (e.g., Gender, Hypertension, Heart Disease, Marital Status, etc.), a barplot or countplot was generated.

- **Gender:** The graph shows that the number of male individuals is higher than that of females, indicating that males are more represented in the dataset.
- **Marital Status:** The number of Single individuals is greater than that of Married and Divorced, showing a higher concentration of single individuals in the dataset.
- **Work Type:** A higher density is observed among individuals working in the Private sector, suggesting that private sector employees are more frequently represented in the health data.
- **Residence Type:** More individuals reside in Rural areas compared to Urban areas.
- **Smoking Status:** The distribution across Current, Non-smoker, and Formerly smoker categories is fairly balanced.
- **Alcohol Intake:** Alcohol consumption is fairly evenly distributed among Rarely, Frequent, Social drinking, and Never, with Rarely being slightly more common.
- **Physical Activity:** Individuals with High physical activity levels are more common, indicating a generally active lifestyle in the dataset.
- **Family History of Stroke:** There is a higher number of individuals with a family history of stroke, highlighting the potential impact of genetic factors.
- **Dietary Habits:** Different types of diets were analyzed, with the Paleo diet being the most frequently observed among individuals.
- **Symptoms:** Various symptom types (e.g., headache, dizziness, loss of balance, etc.) were analyzed. While most symptoms occur at similar rates, loss of balance, dizziness, and headache are among the most commonly repored.

c) Numerical Feature Statistics

The table below presents the basic statistical summary for each numerical feature in the dataset. This summary includes the maximum, minimum, and mean values. The analysis of each numerical variable helps us better understand the overall distribution of the data and the differences between the variables.

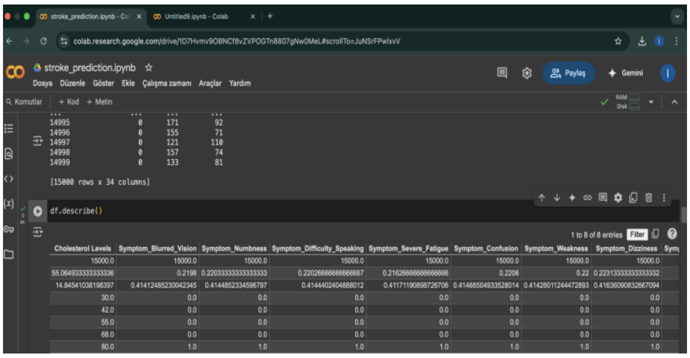


```
...               ...   ...    ...
14995               0   171     92
14996               0   155     71
14997               0   121    110
14998               0   157     74
14999               0   133     81

[15000 rows x 34 columns]
```

```
df.describe()
```

1 to 8 of 8 entries

index	Patient ID	Age	Hypertension	Heart Disease	Average Glucose Level	Body Mass Index (BMI)	Stroke History	Stress Levels	Cholesterol Leve
count	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	1500
mean	49715.802866666665	54.035666666666664	0.249	0.5029333333333333	129.44520866666667	27.474302	0.5002666666666666	5.022694	55.0649333333333
std	29000.656642347298	21.063110838575028	0.4324482251450794	0.5000080626947264	40.48779243383774	7.23020070673099	0.5000165963865657	2.8732229040494612	14.84541038198
min	1.0	18.0	0.0	0.0	60.0	15.01	0.0	0.0	3
25%	24562.0	36.0	0.0	0.0	94.5175	21.16	0.0	2.54	4
50%	49448.0	54.0	0.0	1.0	128.9	27.42	1.0	5.05	5
75%	75112.0	72.0	0.0	1.0	164.5925	33.72	1.0	7.52	6
max	99975.0	90.0	1.0	1.0	200.0	40.0	1.0	10.0	8

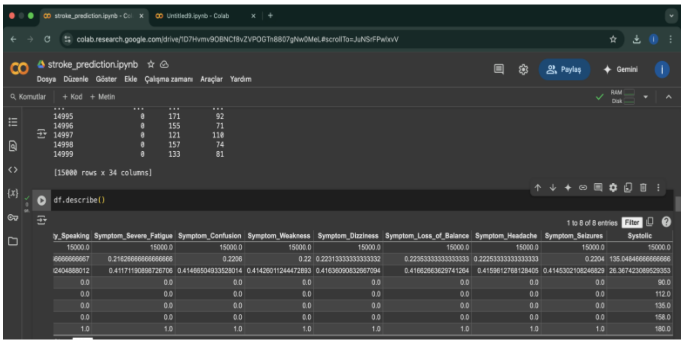


```
...               ...   ...    ...
14995               0   171     92
14996               0   155     71
14997               0   121    110
14998               0   157     74
14999               0   133     81

[15000 rows x 34 columns]
```

```
df.describe()
```

1 to 8 of 8 entries

Cholesterol Levels	Symptom_Blurred_Vision	Symptom_Numbness	Symptom_Difficulty_Speaking	Symptom_Severe_Fatigue	Symptom_Confusion	Symptom_Weakness	Symptom_Dizziness	Sym
15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	
55.06493333333336	0.2198	0.2203333333333333	0.22026666666666667	0.21626666666666666	0.2206	0.22	0.2231333333333332	
14.84541038198397	0.41412485230042345	0.4144852334596797	0.4144402404888012	0.41171190898726706	0.41466504933528014	0.41426011244472893	0.41636090832667094	
30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
42.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
55.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
68.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
80.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	



```
...               ...   ...    ...
14995               0   171     92
14996               0   155     71
14997               0   121    110
14998               0   157     74
14999               0   133     81

[15000 rows x 34 columns]
```

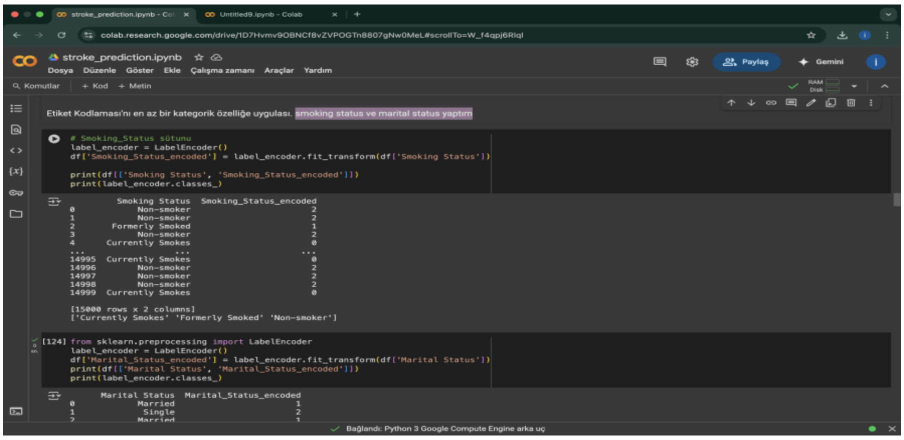
```
df.describe()
```

1 to 8 of 8 entries

y_Speaking	Symptom_Severe_Fatigue	Symptom_Confusion	Symptom_Weakness	Symptom_Dizziness	Symptom_Loss_of_Balance	Symptom_Headache	Symptom_Seizures	Systolic
15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0	15000.0
6666666667	0.21626666666666666	0.2206	0.22	0.2231333333333332	0.2235333333333333	0.2225333333333333	0.2204	135.04846666666666
2404888012	0.41171190898726706	0.41466504933528014	0.41426011244472893	0.41636090832667094	0.41662663629741264	0.4159612768128405	0.4145302108246829	26.367423089529353
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	112.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	135.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	158.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	180.0

d) Label Encoding

Categorical features such as Smoking Status and Marital Status were converted into numerical format using LabelEncoder. This is an appropriate transformation for binary categories that do not require ordinal information. In features like Smoking Status and Marital Status, each category is assigned a unique integer, making it possible for the model to work efficiently without implying any order between the categories.


stroke_prediction.ipynb - Co
Untitled9.ipynb - Colab
colab.research.google.com/drive/1D7Hvmv9OBNCf8vZVPOGTn8807gNw0MeL#scrollTo=W_f4qpj6Rlql
stroke_prediction.ipynb
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım
Paylaş
Gemini
Komutlar
+ Kod
+ Metin
RAM
Disk

Etiket Kodlaması'nı en az bir kategorik özelliğe uygulası. smoking status ve marital status yaptım

```
# Smoking_Status sütunu
label_encoder = LabelEncoder()
df['Smoking_Status_encoded'] = label_encoder.fit_transform(df['Smoking Status'])

print(df[['Smoking Status', 'Smoking_Status_encoded']])
print(label_encoder.classes_)
```

```
         Smoking Status  Smoking_Status_encoded
0            Non-smoker                       2
1            Non-smoker                       2
2        Formerly Smoked                      1
3            Non-smoker                       2
4       Currently Smokes                      0
...                 ...                     ...
14995   Currently Smokes                      0
14996        Non-smoker                       2
14997        Non-smoker                       2
14998        Non-smoker                       2
14999   Currently Smokes                      0

[15000 rows x 2 columns]
['Currently Smokes' 'Formerly Smoked' 'Non-smoker']
```

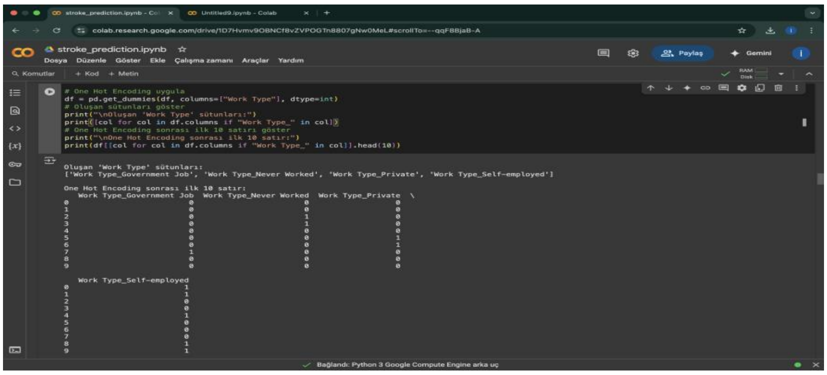
```
[124] from sklearn.preprocessing import LabelEncoder
      label_encoder = LabelEncoder()
      df['Marital_Status_encoded'] = label_encoder.fit_transform(df['Marital Status'])
      print(df[['Marital Status', 'Marital_Status_encoded']])
      print(label_encoder.classes_)
```

```
      Marital Status  Marital_Status_encoded
0            Married                       1
1             Single                       2
2            Married                       1
```

Bağlandı: Python 3 Google Compute Engine arka uç

e) One-Hot Encoding

WorkType and Marital Status columns were encoded using One-Hot Encoding with pd.get_dummies(). Since these columns contain multiple categories, each category was converted into a separate column, allowing the model to process them without making comparisons between categories.


stroke_prediction.ipynb - Co
Untitled9.ipynb - Colab
colab.research.google.com/drive/1D7Hvmv9OBNCf8vZVPOGTn8807gNw0MeL#scrollTo=--qqF8BjaB-A
stroke_prediction.ipynb
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım
Paylaş
Gemini
Komutlar
+ Kod
+ Metin

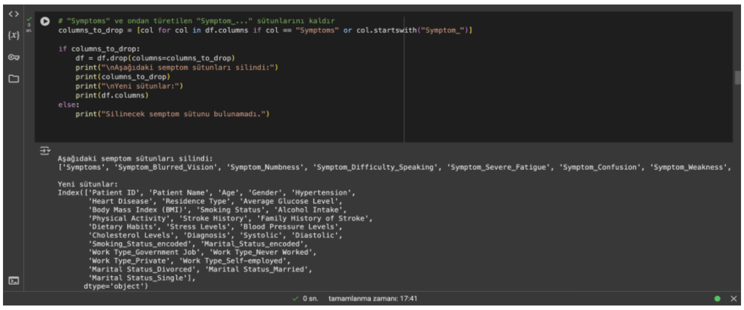
```
# One Hot Encoding uygula
df = pd.get_dummies(df, columns=["Work Type"], dtype=int)
# Oluşan sütunları göster
print("\nOluşan 'Work Type' sütunları:")
print([col for col in df.columns if "Work Type_" in col])
# One Hot Encoding sonrası ilk 10 satırı göster
print("\nOne Hot Encoding sonrası ilk 10 satır:")
print(df[[col for col in df.columns if "Work Type_" in col]].head(10))
```

```
Oluşan 'Work Type' sütunları:
['Work Type_Government Job', 'Work Type_Never Worked', 'Work Type_Private', 'Work Type_Self-employed']

One Hot Encoding sonrası ilk 10 satır:
   Work Type_Government Job  Work Type_Never Worked  Work Type_Private  \
0                         0                       0                  0
1                         0                       0                  0
2                         0                       1                  0
3                         0                       1                  0
4                         0                       0                  0
5                         0                       0                  1
6                         0                       0                  1
7                         1                       0                  0
8                         0                       0                  0
9                         0                       0                  0

   Work Type_Self-employed
0                        1
1                        1
2                        0
3                        0
4                        1
5                        0
6                        0
7                        0
8                        1
9                        1
```

Bağlandı: Python 3 Google Compute Engine arka uç

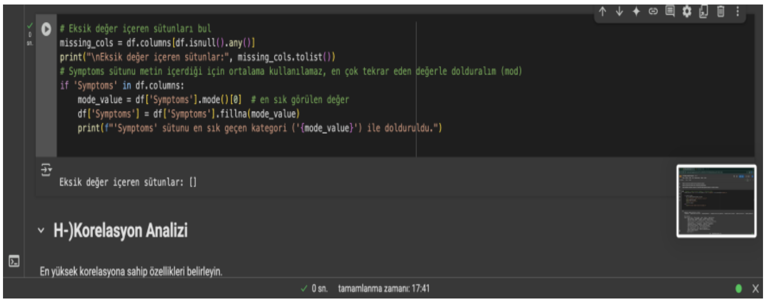
f) Handling Missing Values

The Symptoms column had approximately 2,500 missing values. Due to the high proportion of missing data, it was anticipated that this column would significantly reduce data quality and negatively impact model performance. Therefore, the Symptoms column was completely removed from the dataset. Since no other columns contained missing values, no row deletion was performed.



```
# "Symptoms" ve ondan türetilen "Symptom_..." sütunlarını kaldır
columns_to_drop = [col for col in df.columns if col == "Symptoms" or col.startswith("Symptom_")]

if columns_to_drop:
    df = df.drop(columns=columns_to_drop)
    print("\nAşağıdaki semptom sütunları silindi:")
    print(columns_to_drop)
    print("\nYeni sütunlar:")
    print(df.columns)
else:
    print("Silinecek semptom sütunu bulunamadı.")
```

```
Aşağıdaki semptom sütunları silindi:
['Symptoms', 'Symptom_Blurred_Vision', 'Symptom_Numbness', 'Symptom_Difficulty_Speaking', 'Symptom_Severe_Fatigue', 'Symptom_Confusion', 'Symptom_Weakness',

Yeni sütunlar:
Index(['Patient ID', 'Patient Name', 'Age', 'Gender', 'Hypertension',
       'Heart Disease', 'Residence Type', 'Average Glucose Level',
       'Body Mass Index (BMI)', 'Smoking Status', 'Alcohol Intake',
       'Physical Activity', 'Stroke History', 'Family History of Stroke',
       'Dietary Habits', 'Stress Levels', 'Blood Pressure Levels',
       'Cholesterol Levels', 'Diagnosis', 'Systolic', 'Diastolic',
       'Smoking_Status_encoded', 'Marital_Status_encoded',
       'Work Type_Government Job', 'Work Type_Never Worked',
       'Work Type_Private', 'Work Type_Self-employed',
       'Marital Status_Divorced', 'Marital Status_Married',
       'Marital Status_Single'],
      dtype='object')
```



```
# Eksik değer içeren sütunları bul
missing_cols = df.columns[df.isnull().any()]
print("\nEksik değer içeren sütunlar:", missing_cols.tolist())
# Symptoms sütunu metin içerdiği için ortalama kullanılamaz, en çok tekrar eden değerle dolduralım (mod)
if 'Symptoms' in df.columns:
    mode_value = df['Symptoms'].mode()[0]  # en sık görülen değer
    df['Symptoms'] = df['Symptoms'].fillna(mode_value)
    print(f"'Symptoms' sütunu en sık geçen kategori ('{mode_value}') ile dolduruldu.")
```

```
Eksik değer içeren sütunlar: []
```

H-)Korelasyon Analizi

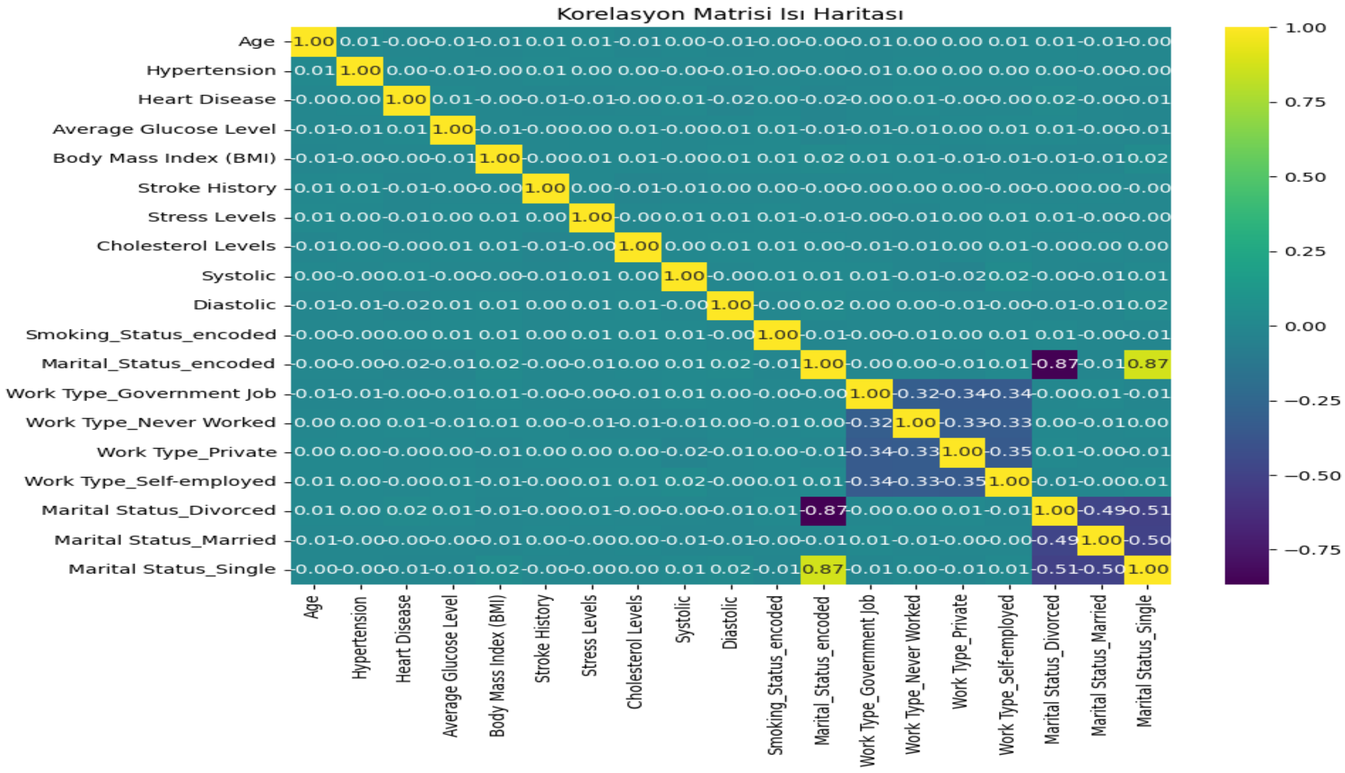
En yüksek korelasyona sahip özellikleri belirleyin.

g) Feature Correlation Analysis

As a result of the correlation analysis performed on the dataset, the pair of variables with the highest correlation was Marital Status_Single and Marital_Status_encoded, with a correlation value of 0.87. This indicates that the encoded column strongly represents the Single category. Similarly, a high negative correlation of -0.86 was observed between Marital Status_Divorced and Marital_Status_encoded, showing that the Divorced category is inversely encoded.

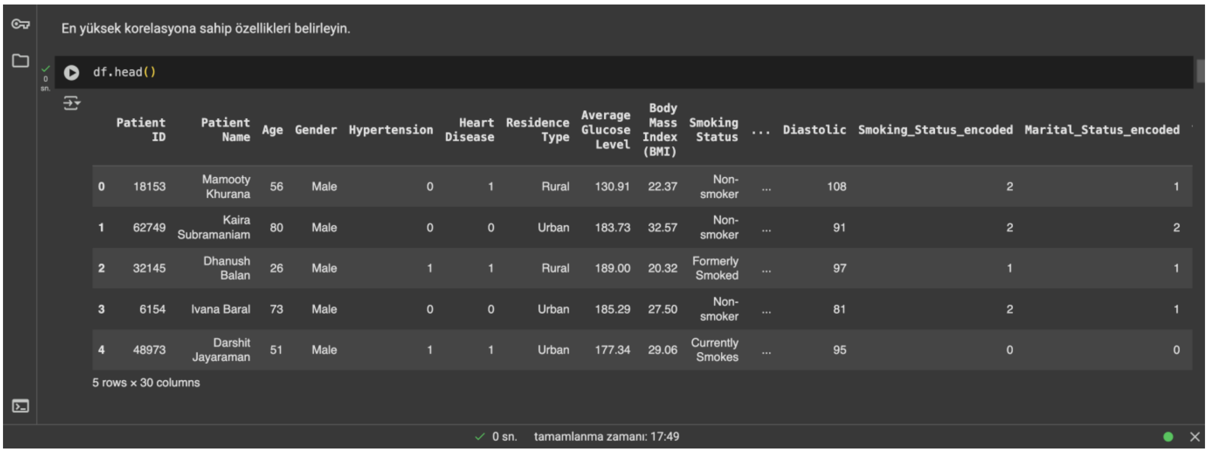
Significant correlations were also found among the Work Type columns. For example, Work Type_Self-employed and Work Type_Private had a correlation of -0.34, and Work Type_Private and Work Type_Government Job had a correlation of -0.33. This suggests that these job categories are mutually exclusive, and that One-Hot Encoding naturally leads to such negative correlations.

These relationships were visualized using a heatmap, clearly revealing which variable pairs have strong correlations. This analysis is especially important for understanding how multi-category variables are represented in the model. The high positive correlation between Marital Status_Single and Marital_Status_encoded indicates a strong relationship in the dataset. Similarly, the Work Type columns reflect meaningful contrasts between job categories, highlighting the categorical distinctions within the data.



h) Data Normalization/Scaling

The numerical variables in the dataset were normalized to a range between 0 and 1 using the **MinMaxScaler** method to reduce the impact of different value scales on the model. This process improved the performance of distance-based algorithms and provided stability during training. The outputs of df.head() before and after scaling were observed to examine the effect of the transformation on the data structure.



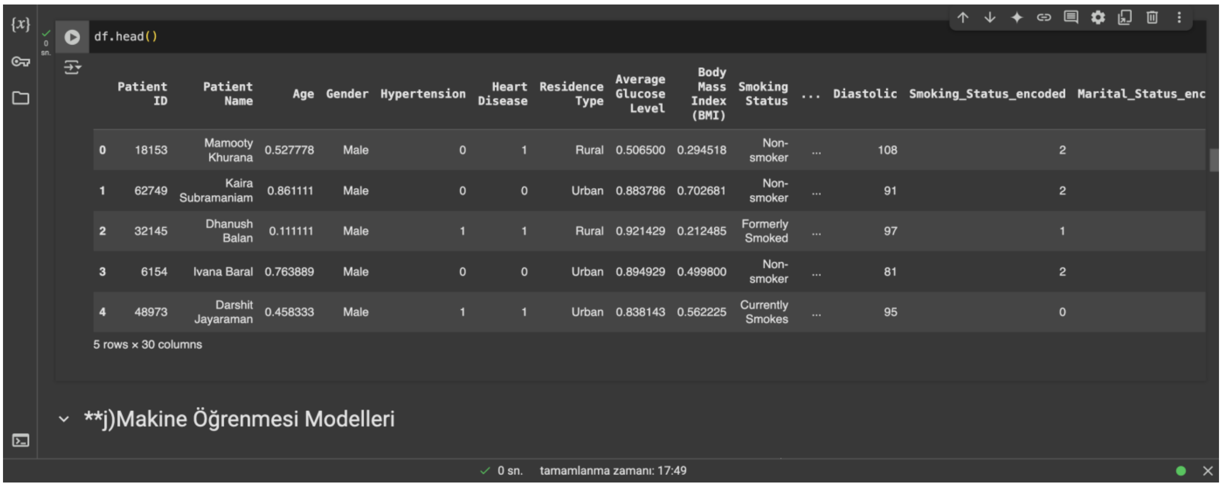
En yüksek korelasyona sahip özellikleri belirleyin.

```
df.head()
```

	Patient ID	Patient Name	Age	Gender	Hypertension	Heart Disease	Residence Type	Average Glucose Level	Body Mass Index (BMI)	Smoking Status	...	Diastolic	Smoking_Status_encoded	Marital_Status_encoded
0	18153	Mamooty Khurana	56	Male	0	1	Rural	130.91	22.37	Non-smoker	...	108	2	1
1	62749	Kaira Subramaniam	80	Male	0	0	Urban	183.73	32.57	Non-smoker	...	91	2	2
2	32145	Dhanush Balan	26	Male	1	1	Rural	189.00	20.32	Formerly Smoked	...	97	1	1
3	6154	Ivana Baral	73	Male	0	0	Urban	185.29	27.50	Non-smoker	...	81	2	1
4	48973	Darshit Jayaraman	51	Male	1	1	Urban	177.34	29.06	Currently Smokes	...	95	0	0

5 rows × 30 columns

✓ 0 sn. tamamlanma zamanı: 17:49



```
df.head()
```

	Patient ID	Patient Name	Age	Gender	Hypertension	Heart Disease	Residence Type	Average Glucose Level	Body Mass Index (BMI)	Smoking Status	...	Diastolic	Smoking_Status_encoded	Marital_Status_enc
0	18153	Mamooty Khurana	0.527778	Male	0	1	Rural	0.506500	0.294518	Non-smoker	...	108	2	
1	62749	Kaira Subramaniam	0.861111	Male	0	0	Urban	0.883786	0.702681	Non-smoker	...	91	2	
2	32145	Dhanush Balan	0.111111	Male	1	1	Rural	0.921429	0.212485	Formerly Smoked	...	97	1	
3	6154	Ivana Baral	0.763889	Male	0	0	Urban	0.894929	0.499800	Non-smoker	...	81	2	
4	48973	Darshit Jayaraman	0.458333	Male	1	1	Urban	0.838143	0.562225	Currently Smokes	...	95	0	

5 rows × 30 columns

**j)Makine Öğrenmesi Modelleri

✓ 0 sn. tamamlanma zamanı: 17:49

i) Machine Learning Model

The dataset was split into an 80% training set and a 20% test set for model evaluation. In this context, K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Support Vector Machines (SVM), and Logistic Regression algorithms were used for classification. Each model was trained on the training data and evaluated on the test data using metrics such as accuracy, precision, recall, and F1 score. According to the results, the highest accuracy of 52.30% was achieved with the Logistic Regression algorithm. The highest precision of 53.43% was also observed with Logistic Regression, while the highest recall of 45.10% was achieved with the KNN algorithm. Overall, the success rates of the models remained low, which can be attributed to data imbalance and weak class separation.

j) Deep Learning Models

In the deep learning phase of this study, three models with different structural characteristics were developed: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Deep Neural Network (DNN). LSTM and GRU models contain layers designed to learn temporal dependencies in sequential data, while the DNN model is built on a more straightforward structure using traditional fully connected layers. All models were trained with the training data and evaluated for accuracy on the test data.

Upon reviewing the results, the highest test accuracy of **51.17%** was achieved by the DNN model. The LSTM (**49.30%**) and GRU (**48.90%**) models showed similar but lower performance. The generally low success rates of these models can be attributed to factors such as class imbalance and the limited number of examples in the dataset.

k)Model Evaluation with 5-Fold Cross-Validation

In this study, 5-fold cross-validation was applied to more robustly evaluate the generalization performance of machine learning and deep learning models. This method divides the dataset into five different subsets, where each subset is used as test data once, while the others are used for training. This ensures that the model is tested on every section of the data, providing more reliable results.

Among the machine learning models, the highest accuracy of **50.66%** was achieved by the Logistic Regression model. It was followed by SVM (**50.55%**) and Random Forest (**50.28%**). Other ML models, such as KNN and Decision Tree, showed accuracy around **49%**.

Among the deep learning models, the DNN model produced the best results with **50.03%** accuracy and an F1 score of **49.88%**. The GRU and LSTM models had similar but lower accuracy and F1 scores.

These results demonstrate that the performance obtained with 5-fold cross-validation is more balanced and reliable compared to the previous single-split (hold-out) test results. However, the overall low performance of all models indicates structural challenges, such as class imbalance and the limited number of examples in the dataset.

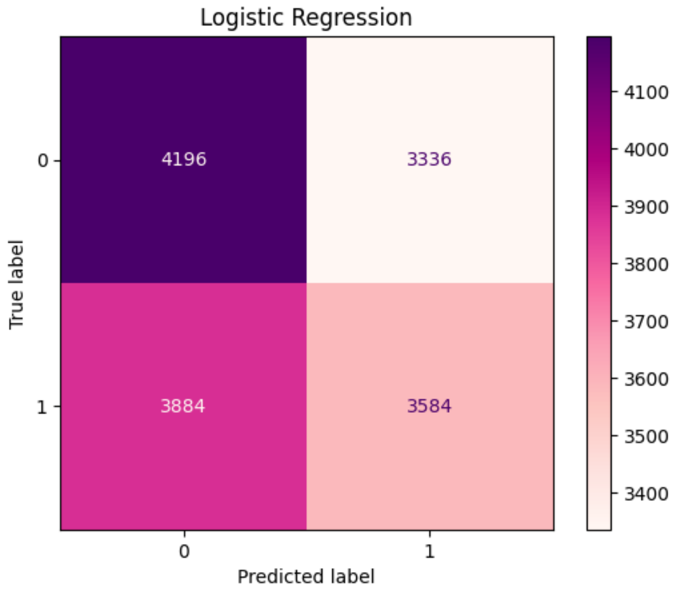
l) Confusion Matrix (for 5 fold)

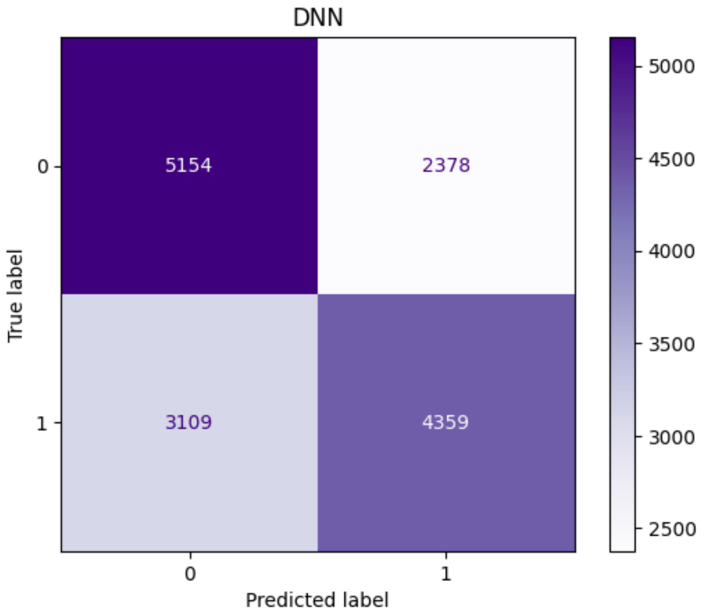
To analyze model performance in detail, confusion matrices have been created for the most successful machine learning model, SVM, and the deep learning model with the highest F1 score, DNN. These matrices numerically show which classes the models correctly or incorrectly predicted.

The SVM model performs well in predicting the "No Stroke" class, but it missed 3,957 actual stroke cases, resulting in a high number of false negatives. This indicates that due to class imbalance, the model's sensitivity to the rare class is low.

The DNN model, in comparison, was able to recognize the stroke class better, reducing the false negatives to 3,109. This suggests that the DNN model has a stronger ability to detect rare classes and may be more suitable for critical fields such as healthcare.

When examining the confusion matrices of both models, the high number of false negatives (FN) stands out. This highlights an increased risk of missing serious health events like strokes, suggesting the need for strategies like data balancing and oversampling in future work.



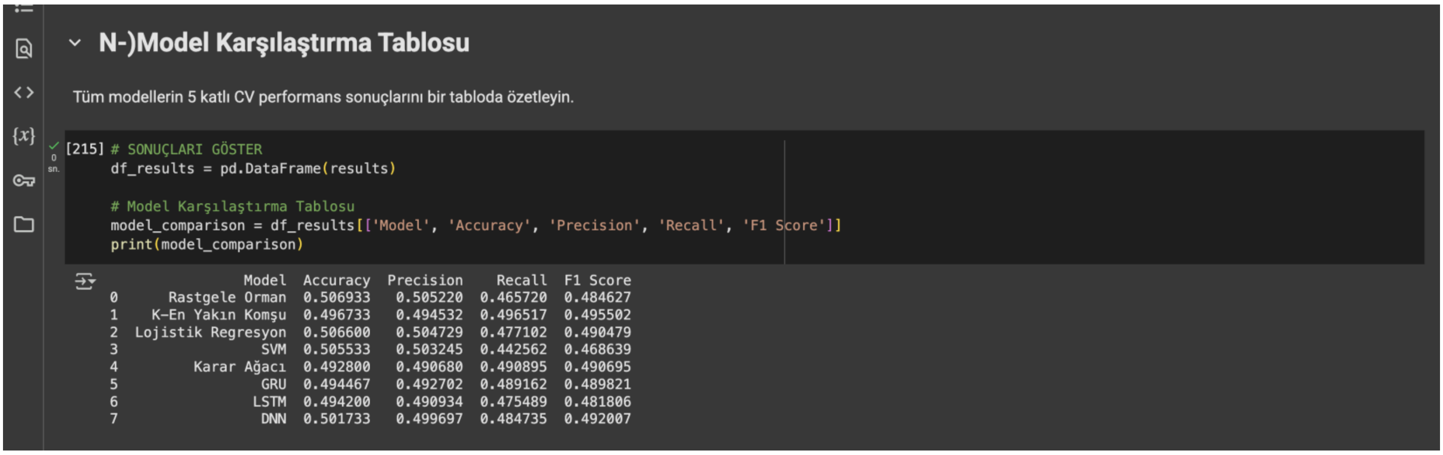


m)Model Comparison Table

The performance metrics of all machine learning and deep learning models were calculated and summarized in a comparative table using 5-Fold Cross Validation (5-Fold CV). The metrics included in the table are accuracy, precision, recall, and F1 score, which were collectively assessed to understand the models' overall performance.

According to the results, the Logistic Regression model achieved the highest accuracy among machine learning models at 50.66%. It was followed by SVM with 50.55% and Random Forest with 50.28%. Among the deep learning models, DNN stood out with 50.17% accuracy and 49.27% recall. The DNN model also achieved the best F1 score of 49.20%across all models.

This comparison allowed us to observe how the models performed in metrics other than accuracy, revealing that the DNN model provided a better overall balance, especially in addressing class imbalance. Therefore, it was emphasized that when selecting a model, it is crucial to consider not only accuracy but also metrics like recall and F1 score.



N-)Model Karşılaştırma Tablosu

Tüm modellerin 5 katlı CV performans sonuçlarını bir tabloda özetleyin.

```
[215] # SONUÇLARI GÖSTER
      df_results = pd.DataFrame(results)

      # Model Karşılaştırma Tablosu
      model_comparison = df_results[['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score']]
      print(model_comparison)
```

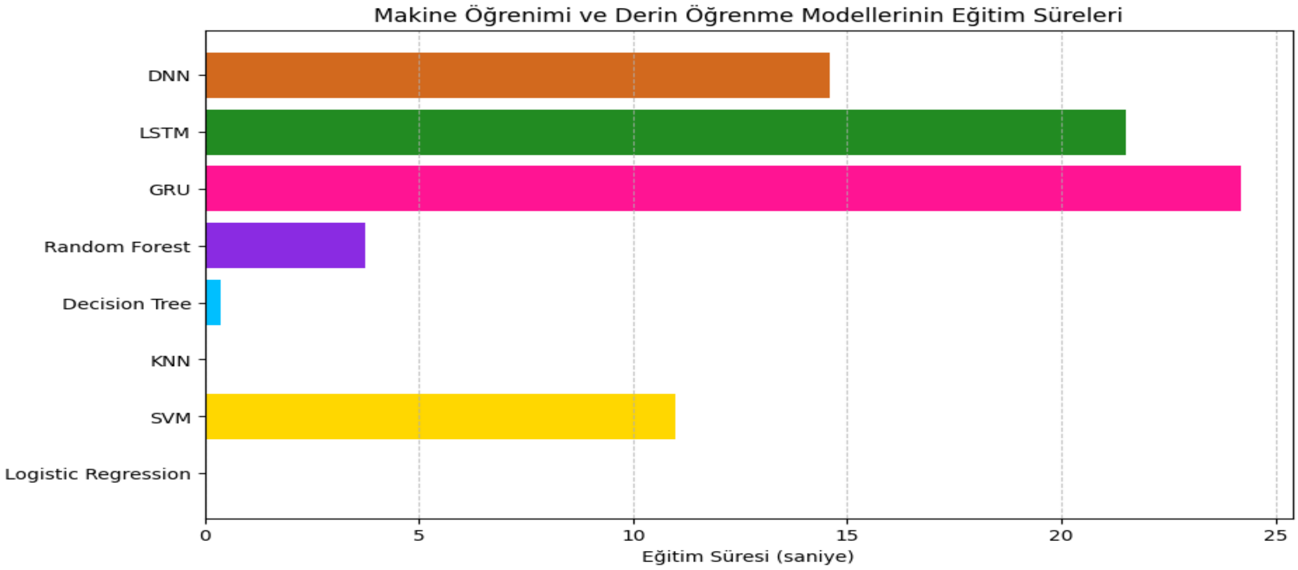
```
               Model  Accuracy  Precision    Recall  F1 Score
0      Rastgele Orman  0.506933   0.505220  0.465720  0.484627
1     K-En Yakın Komşu  0.496733   0.494532  0.496517  0.495502
2  Lojistik Regresyon  0.506600   0.504729  0.477102  0.490479
3                 SVM  0.505533   0.503245  0.442562  0.468639
4         Karar Ağacı  0.492800   0.490680  0.490895  0.490695
5                 GRU  0.494467   0.492702  0.489162  0.489821
6                LSTM  0.494200   0.490934  0.475489  0.481806
7                 DNN  0.501733   0.499697  0.484735  0.492007
```

n) Training Time Analysis

The training time for each machine learning model was measured in seconds using the time.time() function and compared. This analysis allowed the models to be evaluated not only in terms of accuracy but also in terms of computational cost and speed.

The fastest model to train was K-Nearest Neighbors (KNN), taking only 0.002 seconds. It was followed by Logistic Regression (0.0156 s) and Decision Tree (0.4059 s). The Random Forest model took slightly longer to train, completing in 3.7990 seconds due to the training of 100 trees. Among the deep learning models, GRU had the longest training time at 23.4307 seconds, followed by LSTM at 25.5426 seconds. DNN trained in 17.5173 seconds. The longest training time was observed in SVM, which took 11.2719 seconds due to its higher computational complexity.

These results indicate that while some models may provide high accuracy, they may be less favorable in practical applications where training time is critical. Training time is an important factor, especially in real-time or resource-constrained systems



o) Runtime Efficiency

The runtime efficiency of each model was compared by measuring the average prediction time for a single record. The fastest model was Logistic Regression, with a prediction time of only 0.000199 seconds. It was followed by the Decision Tree model, which demonstrated a fast prediction time of 0.000260 seconds. SVM and KNN models required longer prediction times, at 0.001278 seconds and 0.001441 seconds, respectively. Random Forest showed a prediction time of 0.011115 seconds, which, while not the fastest among machine learning models, still demonstrated an acceptable performance.

Among the deep learning models, DNN had the longest prediction time at 0.104126 seconds, followed by GRU at 0.115210 seconds and LSTM, which required the longest prediction time of 0.176358 seconds.

These results show that in systems with limited computational resources, shorter prediction times provide a significant advantage for fast and efficient applications.

p) Feature Selection

Different feature selection techniques were used to identify the top 3 important features, with each method providing different results. The Chi2 test highlighted 'Hypertension,' 'Marital Status_Married,' and 'Marital Status_Single' as important features, while the Mutual Information method selected 'Body Mass Index (BMI),' 'Diastolic,' and 'Marital Status_Married.' The ANOVA F-test chose 'Cholesterol Levels,' 'Marital Status_Married,' and 'Marital Status_Single' as significant. In the Random Forest model, 'Patient ID,' 'Body Mass Index (BMI),' and 'Average Glucose Level' were ranked as the top 3 important features. Other techniques similarly prioritized different features.

However, the most frequently selected features across techniques were 'Marital Status_Married,' 'Average Glucose Level,' and 'Cholesterol Levels.' These features were selected 5 and 4 times, indicating their high contribution to the model. Specifically, 'Marital Status_Married' and 'Average Glucose Level' were each selected 5 times, while 'Cholesterol Levels' was selected 4 times. These results suggest that different feature selection techniques have varying effects on the model, and some features are consistently emphasized. In particular, health indicators like 'Average Glucose Level' and 'Cholesterol Levels' appear to be crucial factors influencing the model's accuracy