



TRABAJO DE FIN DE MÁSTER

Evaluación del riesgo de diabetes/prediabetes a través de un modelo de machine learning

FEBRERO, 2024

ÍNDICE

Evaluación del riesgo de diabetes/prediabetes a través de un modelo de machine learning	3
Introducción	3
Presentación de los datos	3
Análisis Exploratorio de Datos (EDA)	4
Preprocesamiento de datos	8
Selección de variables	8
Búsqueda de mejores hiperparámetros	10
Análisis y selección del mejor modelo	11
Productivización del modelo	13
Referencias	14
Anexos	15
Diccionario de datos	15
¿Cómo está compuesto el conjunto de datos?	21
Variables cuantitativas	25
Variables cualitativas	27
Análisis bivalente: variables cuantitativas	38
Análisis bivalente: variables cualitativas	39
Paso a paso: creación de servicio web usando FASTAPI	58
Paso a paso: despliegue en AMAZON EC2	67
Códigos y video	Error! Bookmark not defined.

EVALUACIÓN DEL RIESGO DE DIABETES/PREDIABETES A TRAVÉS DE UN MODELO DE MACHINE LEARNING

INTRODUCCIÓN

La diabetes es una "una enfermedad crónica (de larga duración) que afecta la forma en que el cuerpo convierte los alimentos en energía." ([Centros para el Control y la Prevención de Enfermedades](#)) Según la Organización Mundial de la Salud, cerca de 422 millones de personas han desarrollado diabetes. Esto supone una carga para el sistema de salud especialmente en países de ingresos medios a bajos. Por ello es importante contar con herramientas que puedan anticipar el desarrollo de diabetes para tomar medidas preventivas. Así, el principal objetivo de este trabajo es la construcción de un modelo de machine learning que permita predecir la probabilidad de que una persona desarrolle prediabetes o diabetes. Además, se pretende llevar a cabo la productivización del modelo mediante la creación de una aplicación web. Esta aplicación facilitará a los usuarios la evaluación de su riesgo de desarrollar esta condición mediante la introducción de información médica, de estilo de vida y demográfica.

PRESENTACIÓN DE LOS DATOS

Para el presente trabajo, se utilizó el conjunto de datos "[Diabetes Health Indicators Dataset](#)", el cual ha sido consolidado por Alex Teboul. Este conjunto de datos se basa en la encuesta de 2015 "[Behavioral Risk Factor Surveillance System](#)" realizada por el Centers for Disease Control and Prevention, la agencia nacional de salud pública de Estados Unidos. Contiene información sobre factores de riesgo asociados a la diabetes.

El conjunto consta de 70692 observaciones y 22 variables, siendo "**Diabetes_binary**" la variable objetivo binaria. Para conocer el detalle de las variables contenidas en el conjunto de datos, así como sus respectivas definiciones y tipos de datos, revisar el [diccionario de datos](#).

ANÁLISIS EXPLORATORIO DE DATOS (EDA)

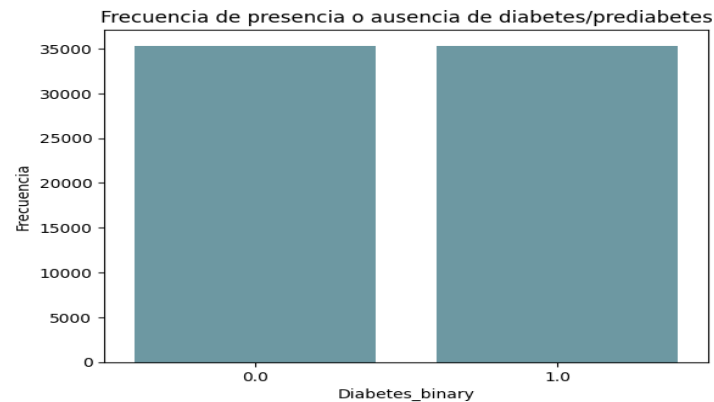
EVALUACIÓN DE LAS VARIABLES

```
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
#   Column              Non-Null Count  Dtype  
---  -
0   Diabetes_binary      70692 non-null  float64
1   HighBP               70692 non-null  float64
2   HighChol             70692 non-null  float64
3   CholCheck            70692 non-null  float64
4   BMI                  70692 non-null  float64
5   Smoker               70692 non-null  float64
6   Stroke               70692 non-null  float64
7   HeartDiseaseorAttack 70692 non-null  float64
8   PhysActivity         70692 non-null  float64
9   Fruits               70692 non-null  float64
10  Veggies              70692 non-null  float64
11  HvyAlcoholConsump    70692 non-null  float64
12  AnyHealthcare        70692 non-null  float64
13  NoDocbcCost          70692 non-null  float64
14  GenHlth              70692 non-null  float64
15  MentHlth             70692 non-null  float64
16  PhysHlth             70692 non-null  float64
17  DiffWalk             70692 non-null  float64
18  Sex                   70692 non-null  float64
19  Age                  70692 non-null  float64
20  Education            70692 non-null  float64
21  Income               70692 non-null  float64
```

- No hay presencia de valores nulos en ninguna de las 22 variables.
- Algunos tipos de datos asociados a ciertas variables son incorrectos y requieren ajustes.
- Se procede a crear una lista con las variables numéricas y una lista con las variables categóricas y asignarles el tipo correcto de dato.
- La variable “BMI” mantiene el tipo de dato float64.
- También, se observa que no es necesario la creación de variables dummies ya que las categorías están representadas por valores binarios.

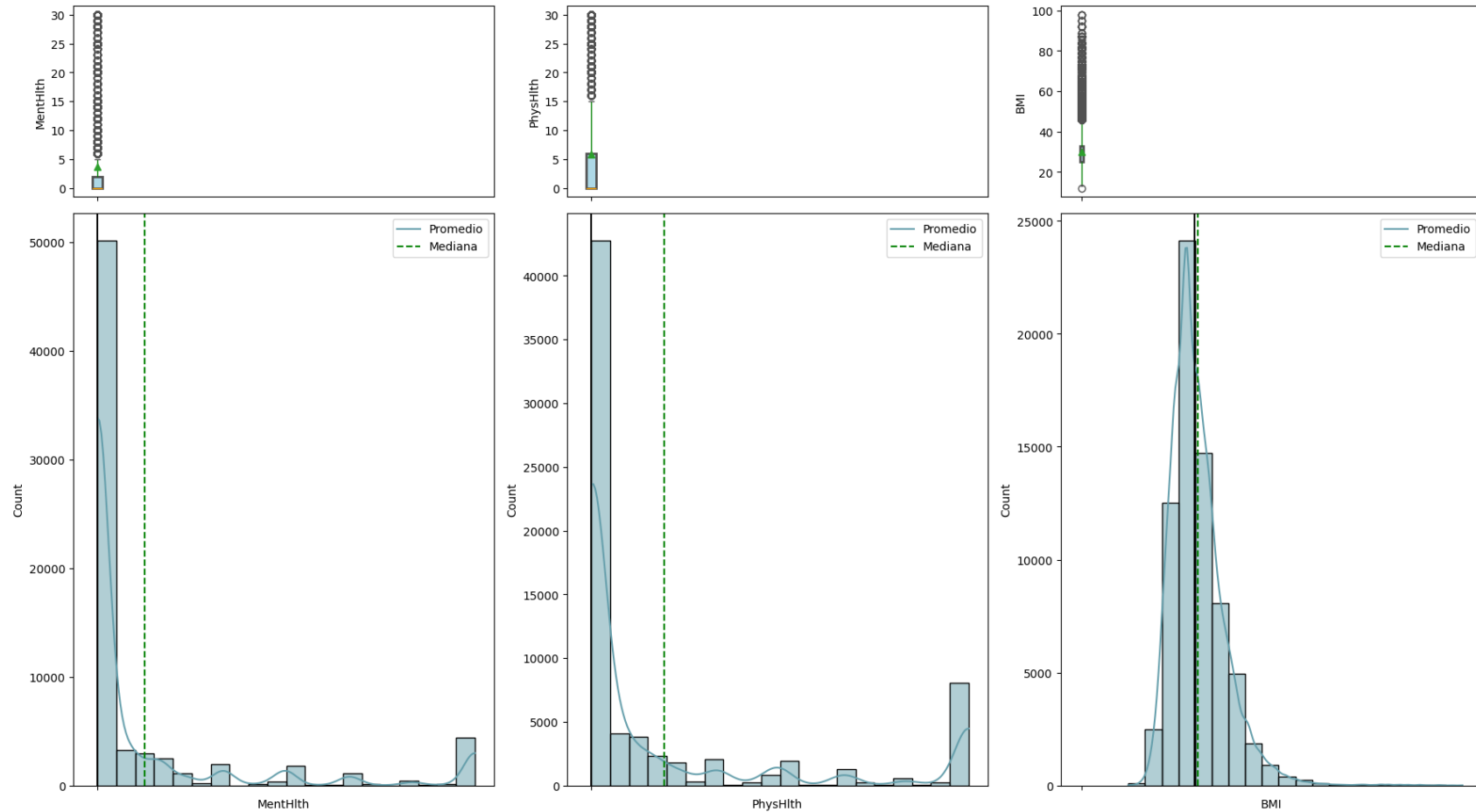
ANÁLISIS DESCRIPTIVO

La variable objetivo “**Diabetes_binary**” presenta la siguiente distribución:



Ambas clases cuentan con el mismo número de observaciones por lo cual estamos ante un conjunto de datos balanceado.

Variables cuantitativas



Se observan valores atípicos para estas tres variables: "MentHlth", "PhysHlth" y "BMI". Sin embargo, teniendo en cuenta que los valores de "MentHlth" y "PhysHlth" siguen una escala que va del 0 al 30, no se tratarán esos valores atípicos.

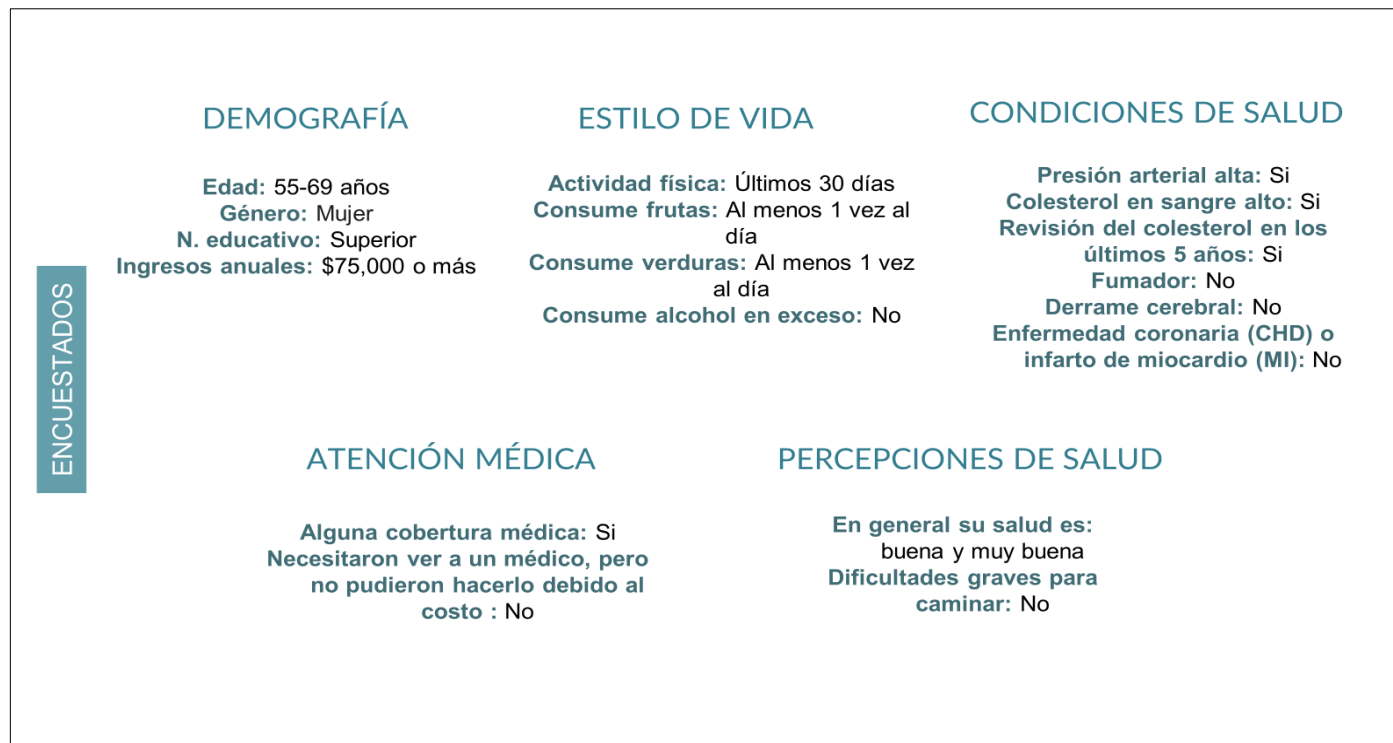
En el caso de "BMI", se observan valores atípicos que parecen ser errores en la entrada de datos. Se han registrado valores de "BMI" mayores a 54, lo cual parece inconsistente. Según la [National Heart, Lung, and Blood Institute \(NIH\)](#), los valores de BMI mayores a 35 generalmente llegan hasta 54. Basándonos en esa información estos valores atípicos deben ser tratados en el paso de preprocesamiento.

Podemos decir además que para la variable:

- "MentHlth": Más de la mitad de los encuestados reportaron pocos días en los que sintieron que su salud mental no fue buena.
- "PhysHlth": Más de la mitad de los encuestados reportaron pocos días en los que sintieron que su salud física no fue buena.
- "BMI": La mayor proporción de los encuestados presentan un BMI menor o igual a 29 lo que los coloca en el rango de peso "normal" y "sobrepeso".

Variables cualitativas

Del análisis de las variables cualitativas ([ver anexo](#)) se desprenden las siguientes características:



ANÁLISIS BIVARIANTE

Para obtener una mejor comprensión del conjunto de datos, se examinan la variable objetivo y las variables cuantitativas y cualitativas. Esto nos permitirá visualizar la existencia de patrones, de correlación y si alguna de las variables impacta en la variable objetivo.

Variables cuantitativas

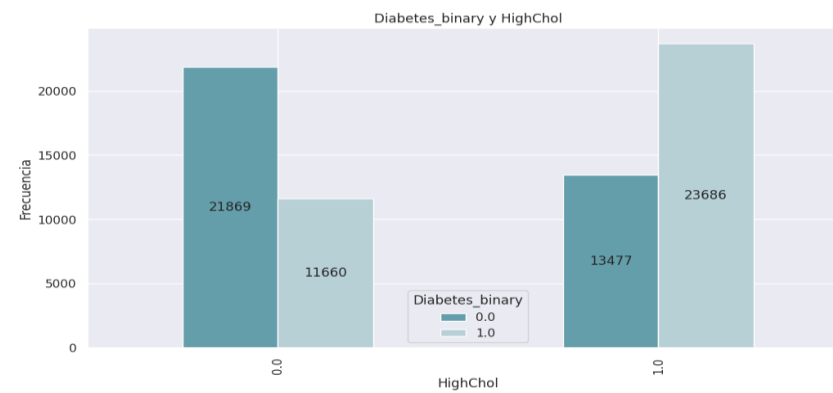
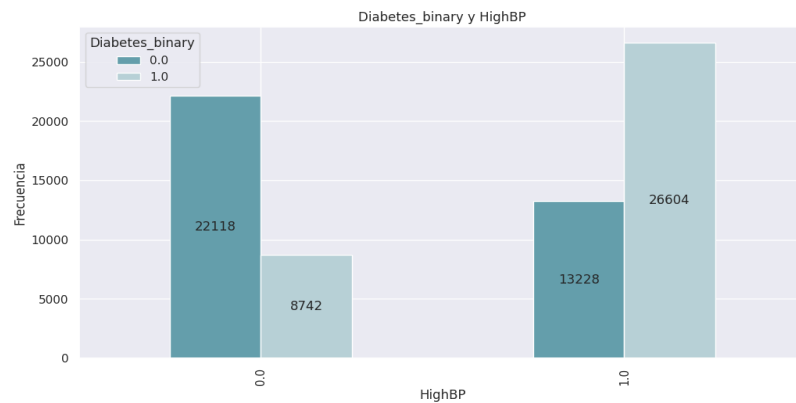
Diabetes_binary	MentHlth	PhysHlth?	BMI
0	3.042268	3.666355	27.76996
1	4.461806	7.954479	31.944011

Se observa que en promedio las personas que presentan diabetes/prediabetes reportan:

- más días en los que su salud mental no fue buena.
- más días en los que su salud física no fue buena.
- un BMI más alto.

Variables cualitativas

Para evaluar si existe una relación significativa entre la variable objetivo y las 18 variables cualitativas se procedió a usar gráficos de barras, y la aplicación de la prueba de chi cuadrado. Se concluye que todas las variables cualitativas presentan una relación significativa con la variable objetivo. La totalidad de los gráficos, así como los valores de la prueba de chi cuadrado pueden verse en el [anexo](#).



PREPROCESAMIENTO DE DATOS

Como señalado en el apartado “Análisis descriptivo: variables cuantitativas”, se deben tratar los outliers para la variable "BMI". Primero, calcularemos la cantidad de observaciones con "BMI" mayor a 54 obteniendo:

Outliers	
Total	Porcentaje
524	0.74%

Dado que el total de outliers no supera el 1%, procederemos a eliminarlos y examinar la nueva distribución de clases:

Nueva distribución	
0	35208
1	34960

Se observa un leve desbalanceo entre las clases por lo que vamos a entrenar nuestros algoritmos con esta distribución y evaluaremos la capacidad de generalizar de los modelos.

SELECCIÓN DE VARIABLES

Trabajaremos con cinco algoritmos: Logistic Regression, Linear SVC, Decision Tree, Random Forest y XGBoost ya que nos enfrentamos a un problema de clasificación binaria. Para los cuatro primeros usaremos RFECV, una técnica de selección de características que elimina de manera iterativa aquellas que no son relevantes. Para XGBoost usaremos SelectFromModel ya que contamos con variables categóricas.

A continuación, un cuadro resumen con los resultados de la selección de variables para cada algoritmo:

Algoritmo	Técnica de selección	Nº óptimo de variables	Variables seleccionadas	Accuracy
Logistic Regression	RFECV	16	BMI', 'HighBP', 'HighChol', 'CholCheck', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'GenHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income'	0.7476
Linear SVC	RFECV	12	HighBP', 'HighChol', 'CholCheck', 'HeartDiseaseorAttack', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'GenHlth', 'DiffWalk', 'Sex', 'Age', 'Income'	0.7322
Decision Tree	RFECV	3	BMI', 'HighBP', 'GenHlth'	0.7208
Random Forest	RFECV	21	MentHlth', 'PhysHlth', 'BMI', 'HighBP', 'HighChol', 'CholCheck', 'Smoker', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income'	0.7356
XGBoost	SelectFromModel	12	HighBP_0.0', 'HighChol_0.0', 'CholCheck_0.0', 'HeartDiseaseorAttack_0.0', 'GenHlth_1.0', 'GenHlth_2.0', 'DiffWalk_0.0', 'Age_1.0', 'Age_2.0', 'Age_3.0', 'Age_4.0', 'Income_8.0'	0.7283

BÚSQUEDA DE MEJORES HIPERPARÁMETROS

Probaremos a buscar los valores óptimos de los hiperparámetros para cada modelo con la finalidad de verificar si estos pueden mejorar su capacidad de generalización. Además, para manejar el leve desbalanceo de clases, ajustaremos el peso de clase óptimo a través del hiperparámetro **class_weight** en los cuatro primeros modelos. Para el caso de XGBoost se ajustó el hiperparámetro **scale_pos_weight** para abordar el desbalanceo de clases.

A continuación, un cuadro resumen con los resultados de la búsqueda:

Modelo	Mejores hiperparámetros	Accuracy
Logistic Regression	C: 0.1 'class_weight': 'balanced' 'dual': False 'penalty': 'l2' 'solver': 'liblinear'	0.7489
Linear SVC	C: 0.1 'class_weight': None 'dual': False 'loss': 'squared_hinge' 'penalty': 'l1'	0.7350
Decision Tree	class_weight: 'balanced' 'criterion': 'gini' 'max_depth': None 'min_samples_leaf': 2 'min_samples_split': 10	0.7258
Random Forest	class_weight: 'balanced' 'max_depth': 10 'min_samples_leaf': 5 'min_samples_split': 2 'n_estimators': 200	0.7509
XGBoost	colsample_bytree: 0.8 'learning_rate': 0.1 'max_depth': 3 'n_estimators': 100 'scale_pos_weight': 1 'subsample': 0.8	0.7332

ANÁLISIS Y SELECCIÓN DEL MEJOR MODELO

Para analizar el rendimiento de los modelos después del ajuste de hiperparámetros, utilizaremos diversas métricas que facilitarán la elección del modelo a usar para el desarrollo del servicio web.

Así tenemos que:

Algoritmo	Accuracy	Accuracy ajuste
Logistic Regression	0.7476	0.7489
Linear SVC	0.7322	0.7350
Decision Tree	0.7208	0.7258
Random Forest	0.7356	0.7509
XGBoost	0.7283	0.7332

- Luego del ajuste de hiperparametros, el accuracy de todos los modelos ha mejorado ligeramente siendo Random Forest el que ha mostrado un incremento superior.

Informe de clasificación

Modelo	Diabetes_binary	Precision	Recall	F1-score	Support
Logistic Regression	0	0.76	0.73	0.74	7045
	1	0.74	0.76	0.75	6989
Linear SVC	0	0.74	0.71	0.73	7045
	1	0.72	0.75	0.74	6989
Decision Tree	0	0.74	0.68	0.71	7045
	1	0.70	0.76	0.73	6989
Random Forest	0	0.78	0.71	0.74	7045
	1	0.73	0.79	0.76	6989
XGBoost	0	0.75	0.68	0.72	7045
	1	0.71	0.77	0.74	6989

- El informe de clasificación muestra que en general, los modelos muestran un rendimiento aceptable.
- Logistic Regression tiende a un mejor rendimiento al predecir casos positivos que Linear SVC. Sin embargo, Random Forest y XGBoost presentan un F1-score más alto para la clase “1” lo que sugiere una mejor capacidad para predecir casos positivos.
- Decision Tree es el modelo que mayor variación entre clases presenta lo que puede indicar que presenta dificultades para mantener un equilibrio entre “Precision” y “Recall” para las dos clases.

Matriz de confusión

Modelo	Verdaderos positivos	Falsos negativos	Falsos positivos	Verdaderos negativos
Logistic Regression	5342	1647	1901	5144
Linear SVC	5252	1737	2016	5029
Decision Tree	5305	1684	2244	4801
Random Forest	5548	1441	2071	4974
XGBoost	5413	1576	2231	4814

- Se observa que los modelos Random Forest y XGBoost presentan un equilibrio sólido entre las clases.
- Logistic Regression y Linear SVC presentan un buen rendimiento en la predicción de verdaderos positivos y verdaderos negativos.
- Decision Tree presenta un número superior de falsos positivos y un número menor de verdaderos negativos lo que muestra una mayor variación de clases como indicado anteriormente.

Al analizar la información obtenida de las métricas:

- **Podemos descartar Decision Tree** ya que este modelo presenta el accuracy más bajo (72%), una mayor variación entre clases y una cantidad significativa de falsos positivos. Además, utiliza únicamente 3 variables para realizar la predicción, lo que sugiere que podemos vernos ante un caso de underfitting.

- También, **descartamos Random Forest** ya que si bien presenta el accuracy más alto (75%), un equilibrio entre las clases y una cantidad significativa de verdaderos positivos y verdaderos negativos. Este modelo es más complejo al utilizar las 21 variables para llegar a ese grado de precisión en su predicción. Ello podría indicarnos que estamos ante un caso de sobreajuste lo que afectaría la capacidad de generalizar sobre nuevos datos.
- Tanto XGBoost como Linear SVC usan 12 variables para realizar la predicción. Sin embargo, el accuracy de Linear SVC es ligeramente superior y es un modelo más simple y con mejor velocidad de entrenamiento.
- Finalmente, **seleccionamos a Logistic Regression como el mejor modelo** a pesar de tener 4 características adicionales en comparación con Linear SVC. Este modelo destaca por ser simple, fácil de interpretar y con una mejor velocidad de entrenamiento en comparación con los otros modelos presentados. Además, logró un accuracy de $\approx 75\%$, y el informe de clasificación muestra un rendimiento equilibrado en términos de precision, recall y f1-score.

Si bien podría esperarse un accuracy mayor para un modelo de predicción de diabetes/prediabetes, es importante tener en cuenta que el conjunto de datos utilizado para el análisis no incluye variables relevantes como "glucosa basal", "insulina", "hemoglobina glicosilada", "grosor de la piel", etc. La falta de estas y otras variables importantes puede limitar la capacidad del modelo para lograr una precisión más alta.

PRODUCTIVIZACIÓN DEL MODELO

El modelo seleccionado (Logistic Regression) se guardó en un archivo pickle. Previamente, se realizaron pruebas en el notebook para corroborar que se pueden obtener predicciones.

Para productivizar el modelo primero, construiremos un servicio web usando el framework FastAPI. Una vez que corroboremos que el servicio funciona de manera local procederemos a alojarlo en una instancia de EC2. Finalmente, realizaremos pruebas adicionales para asegurarnos que el servicio se encuentre en línea y sea accesible para su consumo a través de un formulario web.

Puede encontrar los detalles técnicos de la productivización del modelo en el [anexo](#).

REFERENCIAS

Calcagni, L. (2023, 21 de julio). *Deploy your FastAPI API to AWS EC2 using Nginx*. Medium.
<https://lcalcagni.medium.com/deploy-your-fastapi-to-aws-ec2-using-nginx-aa8aa0d85ec7>

Data Science Duniya. (2021, 18 de abril). *How to deploy machine learning models as a microservice using fastapi*.
<https://ashutoshtripathi.com/2021/02/15/how-to-deploy-machine-learning-models-as-a-microservice-using-fastapi/>

pixegami. (2022, 5 de mayo). *Cómo implementar FastAPI en AWS EC2: ¡pasos rápidos y sencillos!* [Video]. Youtube.
<https://www.youtube.com/watch?v=SgSnz7kW-Ko>

Roby, Eric. (2023, 9 de julio). *Deploy FastAPI on AWS Lambda | In 9 MINUTES* [Video]. Youtube.
https://www.youtube.com/watch?v=7-CvGFJNE_o

scikit-learn. Sklearn.feature_selection.RFECV.
https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

scikit-learn. Sklearn.feature_selection.SelectFromModel.
https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html

scikit-learn. Sklearn.linear_model.LogisticRegression.
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

scikit-learn. Sklearn.svm.LinearSVC.
<https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>

Zhang, Z. (2019, 10 de agosto). *Feature selection why & how explained*. Medium.
<https://towardsdatascience.com/feature-selection-why-how-explained-part-2-352d9130c2e1>

ANEXOS

DICCIONARIO DE DATOS

A continuación, se presentan las variables contenidas en el conjunto de datos, así como sus respectivas definiciones y tipos de datos. Siendo “**Diabetes_binary**” la variable objetivo binaria.

Variable	Explicación	Valores	Tipo
Diabetes_binary	Presencia o ausencia de diabetes/prediabetes	0 = No diabetes	Cualitativa categórica
		1 = Prediabetes y/o diabetes	
HighBP	Adultos a quienes se les ha informado que tienen presión arterial alta por un médico, enfermera u otro profesional de la salud	0 = no presión arterial alta	Cualitativa categórica
		1 = presión arterial alta	
HighChol	¿Alguna vez un médico, enfermero u otro profesional de la salud le ha informado que su colesterol en la sangre está alto?	0 = no colesterol alto	Cualitativa categórica
		1 = colesterol alto	

CholCheck	Evaluación del colesterol en los últimos cinco años	0 = no se ha revisado el colesterol en los últimos 5 años	Cualitativa categórica
		1 = sí se ha revisado el colesterol en los últimos 5 años	
BMI	Índice de Masa Corporal	BMI > = 1	Cuantitativa continua
		Rango 1 - 9999	
Smoker	¿Ha fumado al menos 100 cigarrillos en toda su vida? [Nota: 5 paquetes = 100 cigarrillos]	0 = no	Cualitativa categórica
		1 = sí	
Stroke	(Alguna vez le dijeron) que tuvo un derrame cerebral	0 = no	Cualitativa categórica
		1 = sí	
HeartDiseaseorAttack	Enfermedad coronaria (CHD) o infarto de miocardio (MI)	0 = no	Cualitativa categórica
		1 = sí	
PhysActivity	Actividad física en los últimos 30 días, excluyendo el trabajo	0 = no	Cualitativa categórica
		1 = sí	
Fruits	Consume frutas 1 o más veces al día	0 = no	Cualitativa categórica

		1 = sí	
Veggies	Consume verduras 1 o más veces al día	0 = no	Cualitativa categórica
		1 = sí	
HvyAlcoholConsump	Personas que consumen alcohol en exceso (hombres adultos que tienen más de 14 bebidas por semana y mujeres adultas que tienen más de 7 bebidas por semana)	0 = no	Cualitativa categórica
		1 = sí	
AnyHealthcare	Tiene algún tipo de cobertura de atención médica, incluyendo seguro de salud, planes prepagos como HMO, etc.	0 = no	Cualitativa categórica
		1 = sí	

NoDocbcCost	¿Hubo algún momento en los últimos 12 meses en que necesitó ver a un médico, pero no pudo debido al costo?	0 = no	Cualitativa categórica
		1 = sí	
GenHlth	¿Diría que en general su salud es: escala de 1 a 5	1 = excelente	Cualitativa categórica
		2 = muy buena	
		3 = buena	
		4 = regular	
		5 = mala	
MentHlth	Pensando en su salud mental, que incluye estrés, depresión y problemas emocionales, ¿durante cuántos días de los últimos 30 su salud mental no fue buena?	Escala de 0 - 30 días	Cuantitativa discreta
PhysHlth	Pensando en su salud física, que incluye enfermedades y lesiones físicas, ¿durante cuántos días de los últimos 30 su salud física no fue buena?	Escala de 0 - 30 días	Cuantitativa discreta
DiffWalk	¿Tiene dificultades graves para caminar o subir escaleras?	0 = no	Cualitativa categórica
		1 = sí	
Sex	Sexo	0 = mujer	Cualitativa categórica
		1 = hombre	
Age	Categoría de edad de 13 niveles	1 = 18-24	Cualitativa categórica
		2 = 25-29	

		3 = 30-34 4 = 35-39 5 = 40-44 6 = 45-49 7 = 50-54 8 = 55-59 9 = 60-64 10 = 65-69 11 = 70-74 12 = 75-79 13 = 80 a más	
Education	Nivel educativo, escala de 1 a 6	1 = Nunca asistió a la escuela o solo jardín de infantes 2 = Grados 1 al 8 (Primaria) 3 = Grados 9 al 11 (Algunos años de secundaria) 4 = Grado 12 o GED (Graduado de secundaria) 5 = Universidad 1 a 3 años (Algunos años de universidad o escuela técnica) 6 = Universidad 4 años o más (Graduado universitario)	Cualitativa categórica
Income	Ingresos anuales, escala de 1 a 8	1 = Menos de \$10,000 2 = Menos de \$15,000	Cualitativa categórica

		3 = Menos de \$20,000	4 = Menos de \$25,000	
		5 = Menos de \$35,000	6 = Menos de \$50,000	
		7 = Menos de \$75,000	8 = \$75,000 o más	

CÓDIGO

¿CÓMO ESTÁ COMPUESTO EL CONJUNTO DE DATOS?

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Diabetes_binary        70692 non-null  float64
1   HighBP                 70692 non-null  float64
2   HighChol               70692 non-null  float64
3   CholCheck              70692 non-null  float64
4   BMI                   70692 non-null  float64
5   Smoker                 70692 non-null  float64
6   Stroke                 70692 non-null  float64
7   HeartDiseaseorAttack   70692 non-null  float64
8   PhysActivity           70692 non-null  float64
9   Fruits                 70692 non-null  float64
10  Veggies                70692 non-null  float64
11  HvyAlcoholConsump      70692 non-null  float64
12  AnyHealthcare          70692 non-null  float64
13  NoDocbcCost            70692 non-null  float64
14  GenHlth                70692 non-null  float64
15  MentHlth               70692 non-null  float64
16  PhysHlth               70692 non-null  float64
17  DiffWalk               70692 non-null  float64
18  Sex                    70692 non-null  float64
19  Age                    70692 non-null  float64
20  Education              70692 non-null  float64
21  Income                 70692 non-null  float64
dtypes: float64(22)
memory usage: 11.9 MB
```

- Compuesto de 70692 observaciones y 22 columnas.
- No hay presencia de valores nulos en ninguna de las 22 variables.
- Se observa que algunos tipos de datos asociados a ciertas variables son incorrectos y requieren ajustes.

Procedemos a cambiar el tipo de dato de las variables. Así:

```
#Creamos una lista para las variables numéricas  
var_int = ['MentHlth', 'PhysHlth']  
data[var_int] = data[var_int].astype("int")
```

```
#Creamos una lista para las variables categóricas  
var_cat = ['Diabetes_binary', 'HighBP', 'HighChol', 'CholCheck', 'Smoker', 'Stroke', 'HeartDiseaseorAttack',  
           'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income']  
data[var_cat] = data[var_cat].astype("category")
```

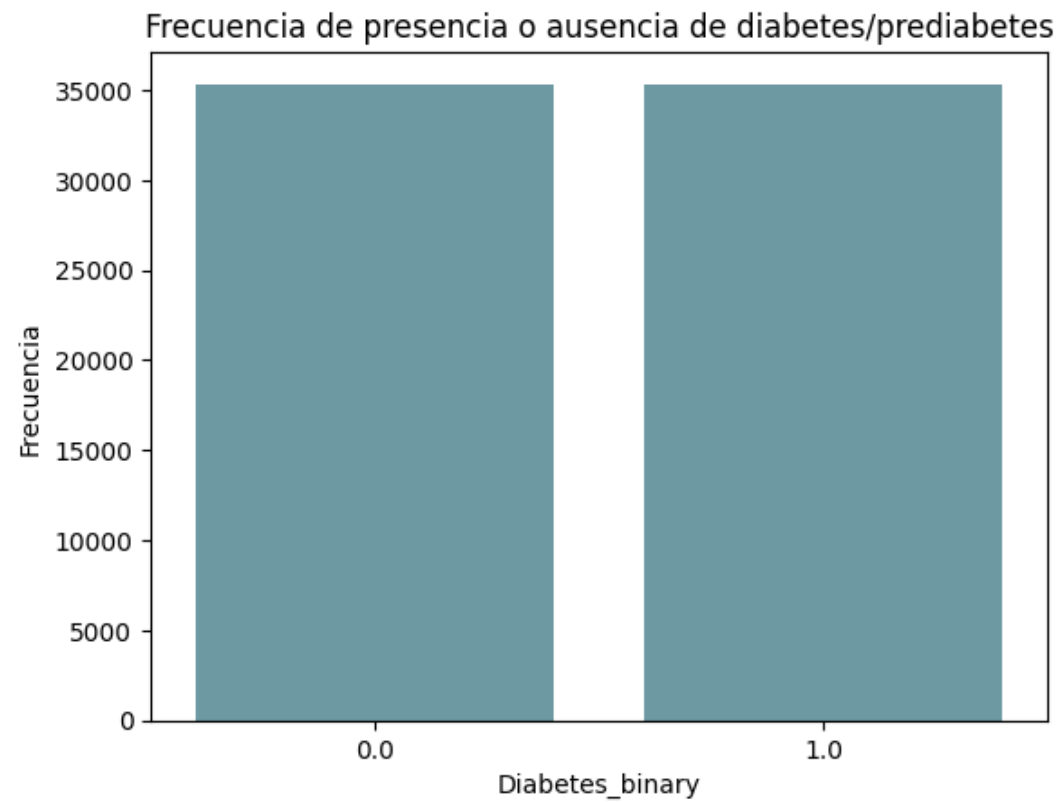
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Diabetes_binary        70692 non-null  category
1   HighBP                 70692 non-null  category
2   HighChol               70692 non-null  category
3   CholCheck              70692 non-null  category
4   BMI                   70692 non-null  float64
5   Smoker                 70692 non-null  category
6   Stroke                 70692 non-null  category
7   HeartDiseaseorAttack   70692 non-null  category
8   PhysActivity           70692 non-null  category
9   Fruits                 70692 non-null  category
10  Veggies                70692 non-null  category
11  HvyAlcoholConsump      70692 non-null  category
12  AnyHealthcare          70692 non-null  category
13  NoDocbcCost            70692 non-null  category
14  GenHlth                70692 non-null  category
15  MentHlth               70692 non-null  int64
16  PhysHlth               70692 non-null  int64
17  DiffWalk               70692 non-null  category
18  Sex                    70692 non-null  category
19  Age                    70692 non-null  category
20  Education               70692 non-null  category
21  Income                 70692 non-null  category
dtypes: category(19), float64(1), int64(2)
memory usage: 2.9 MB
```

Número de observaciones para la variable objetivo

Número de observaciones para la variable objetivo

```
color = '#649EAB'
diabetes_b = data["Diabetes_binary"].value_counts()
sns.barplot(x=diabetes_b.index, y=diabetes_b.values, color = color)
plt.title('Frecuencia de presencia o ausencia de diabetes/prediabetes')
plt.xlabel('Diabetes_binary')
plt.ylabel('Frecuencia')
plt.show()
```



El conjunto de datos está balanceado.

ANÁLISIS EXPLORATORIO DE DATOS (EDA)

VARIABLES CUANTITATIVAS

```
""" Función para calcular estadísticas descriptivas para las variables cuantitativas
x: Nombre de la variable (escribirla con comillas)
var_list: lista que incluye todas las variables del conjunto de datos
"""

def quant_var_stats(x):
    #Definimos el tipo de variable
    var_list = data.columns
    if x in var_list:
        if data[x].dtype == 'object':
            print('\033[1m' + 'Variable Cualitativa (Categórica)' + '\033[0m')
        elif data[x].dtype == 'int64':
            print('\033[1m' + 'Variable Cuantitativa Discreta' + '\033[0m')
        elif data[x].dtype == 'float64':
            print('\033[1m' + 'Variable Cuantitativa Continua' + '\033[0m')

    #Calculamos la media, mediana y moda
    mean=data[x].mean()
    median=data[x].median()
    mode=data[x].mode()
    print('\033[1m' + '\nMedidas de Tendencia Central' + '\033[0m')
    print('Media: ',mean,'\nMediana: ',median,'\nModa: ',mode)

    #Calculamos el Max, Min, Primer cuartil, Tercer cuartil
    max=data[x].max()
    min=data[x].min()
    First_quartile=data[x].quantile(0.25)
    Third_quartile=data[x].quantile(0.75)
    IQR=Third_quartile-First_quartile
    print('\033[1m' + '\nMedidas de Posición' + '\033[0m')
    print('Mínimo: ',min,'\nMáximo: ',max,'\nWhisker Inferior:',First_quartile-1.5*IQR,'\n1er Cuartil: ',First_quartile,'\n3er Cuartil: ',Third_quartile,'\nWhisker Superior:',Third_quartile+1.5*IQR)

    #Calculamos medidas de dispersión
    STD=data[x].std()
    print('\033[1m' + '\nMedidas de variabilidad' + '\033[0m')
    print('Rango: ',max-min,'\nIQR: ',IQR,'\nDesviación estándar: ',STD)
```

```

"""Función para generar boxplot e histograma
feature: dataframe con las columnas a plotear
figsize: tamaño de la figura
bins: número de bins
"""
def histogram_boxplot(feature, figsize=(18, 10), bins=None):
    num_cols = feature.shape[1]

    f, axes = plt.subplots(nrows=2, ncols=num_cols, sharex='col',
                           gridspec_kw={"height_ratios": (.25, .75)},
                           figsize=figsize)

    for i, col in enumerate(feature.columns):
        sns.boxplot(feature[col], ax=axes[0, i], showmeans=True, color= color, medianprops={'color': 'orange'}, boxprops=dict(facecolor='lightblue', linewidth=2), whiskerprops=dict(color='green'))
        sns.histplot(feature[col], ax=axes[1, i], bins=bins, kde=True, color= color) if bins else sns.histplot(feature[col], kde=True, ax=axes[1, i], color='lightcoral')
        axes[1, i].axvline(np.mean(feature[col]), color='green', linestyle='--')
        axes[1, i].axvline(np.median(feature[col]), color='black', linestyle='-')
        axes[1, i].legend({"Promedio": np.mean(feature[col]), "Mediana": np.median(feature[col])})

    plt.tight_layout()
    plt.show()

```

`quant_var_stats('MentHlth')`

Variable Cuantitativa Discreta

Medidas de Tendencia Central

Media: 3.7520370056017653

Mediana: 0.0

Moda: 0 0

Name: MentHlth, dtype: int64

Medidas de Posición

Mínimo: 0

Máximo: 30

Whisker Inferior: -3.0

1er Cuartil: 0.0

3er Cuartil: 2.0

Whisker Superior: 5.0

Medidas de variabilidad

Rango: 30

IQR: 2.0

Desviación estándar: 8.155626553608068

`quant_var_stats('PhysHlth')`

Variable Cuantitativa Discreta

Medidas de Tendencia Central

Media: 5.810417020313473

Mediana: 0.0

Moda: 0 0

Name: PhysHlth, dtype: int64

Medidas de Posición

Mínimo: 0

Máximo: 30

Whisker Inferior: -9.0

1er Cuartil: 0.0

3er Cuartil: 6.0

Whisker Superior: 15.0

Medidas de variabilidad

Rango: 30

IQR: 6.0

Desviación estándar: 10.06226053116389

`quant_var_stats('BMI')`

Variable Cuantitativa Continua

Medidas de Tendencia Central

Media: 29.856985231709388

Mediana: 29.0

Moda: 0 27.0

Name: BMI, dtype: float64

Medidas de Posición

Mínimo: 12.0

Máximo: 98.0

Whisker Inferior: 13.0

1er Cuartil: 25.0

3er Cuartil: 33.0

Whisker Superior: 45.0

Medidas de variabilidad

Rango: 86.0

IQR: 8.0

Desviación estándar: 7.1139538515768415

VARIABLES CUALITATIVAS

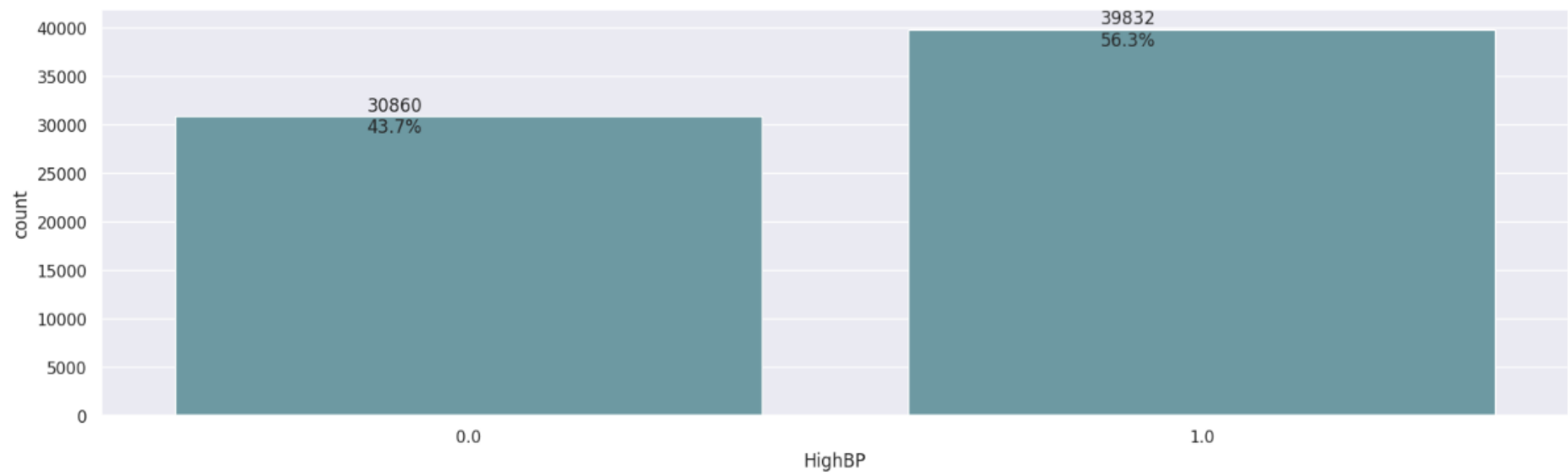
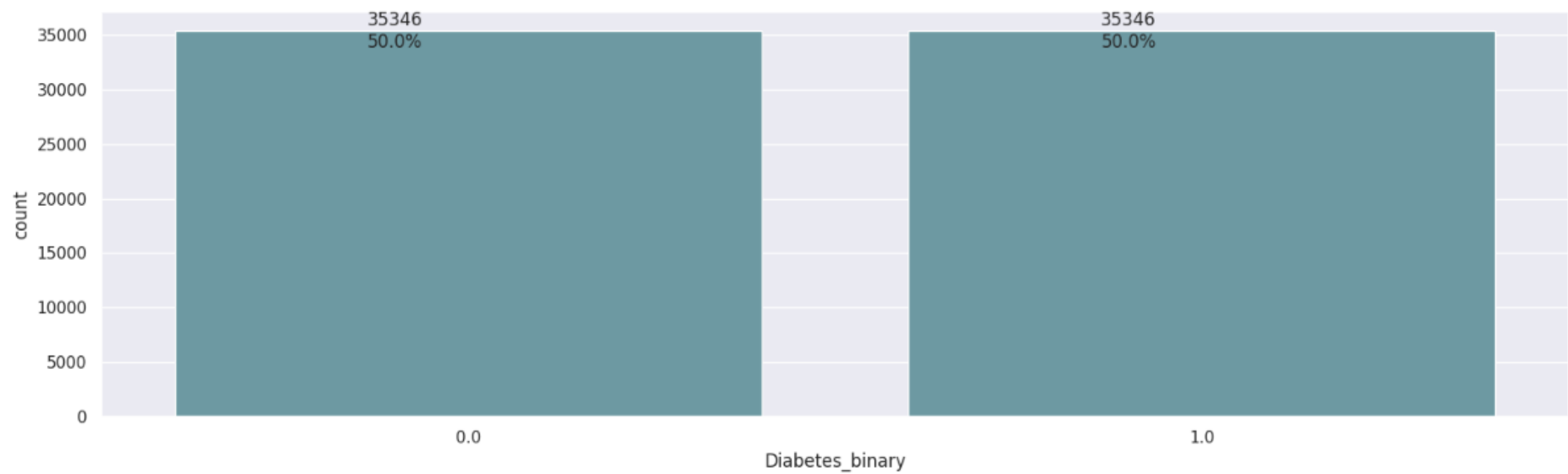
```
def perc_on_bar(feature):
    #Creamos el countplot
    sns.set(rc={'figure.figsize': (18, 5)}) #Ajustamos el ancho y alto
    ax = sns.countplot(x=feature, data=data, palette=['#649EAB'])

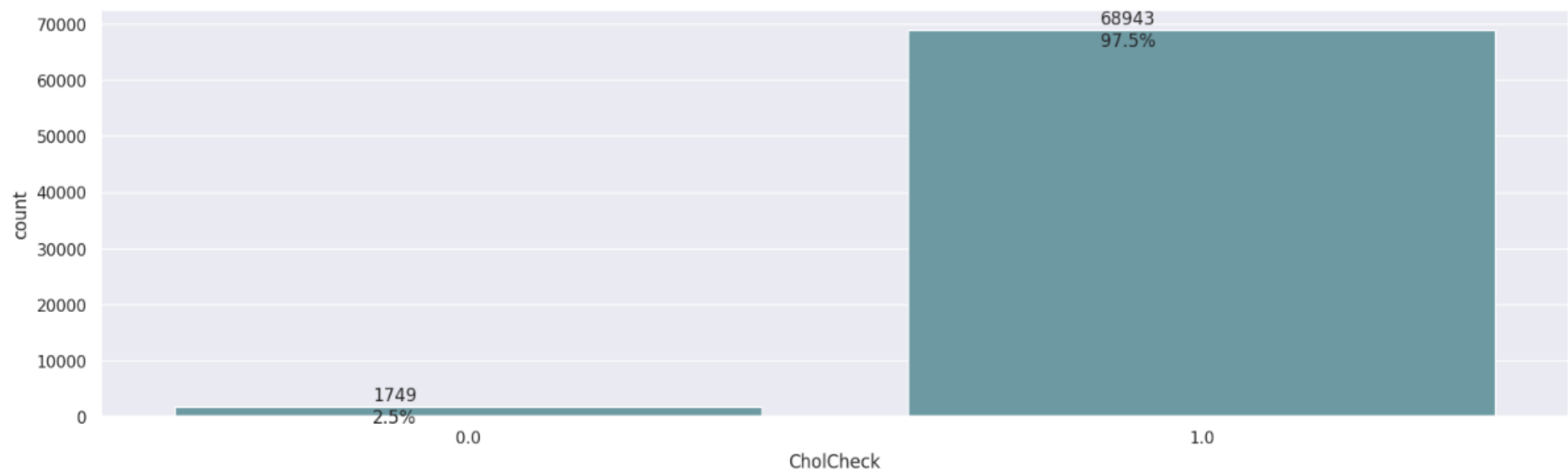
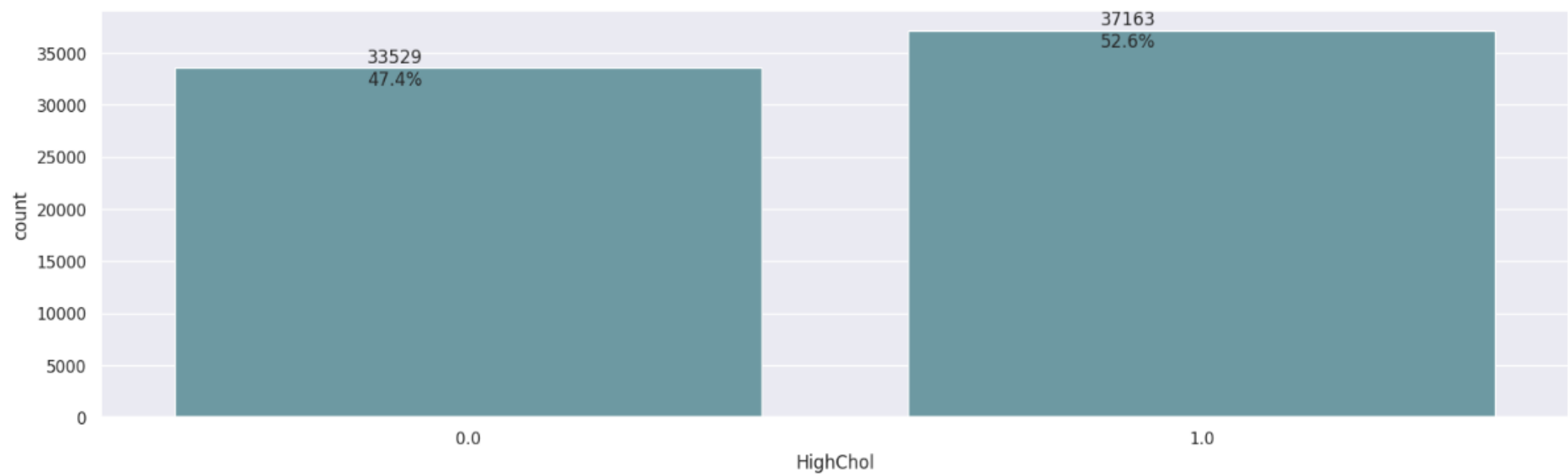
    total = len(data)
    for p in ax.patches:
        count = int(p.get_height()) #Recuento de obs. para las categorías
        percentage = '{:.1f}%'.format(100 * count / total) #Porcentaje de cada clase por categoría
        x = p.get_x() + p.get_width() / 2 - 0.1 #ancho
        y = p.get_y() + p.get_height() #alto
        ax.annotate(f'{count}\n{percentage}', (x, y), ha='center', va='center', size=12) #Recuento y porcentaje

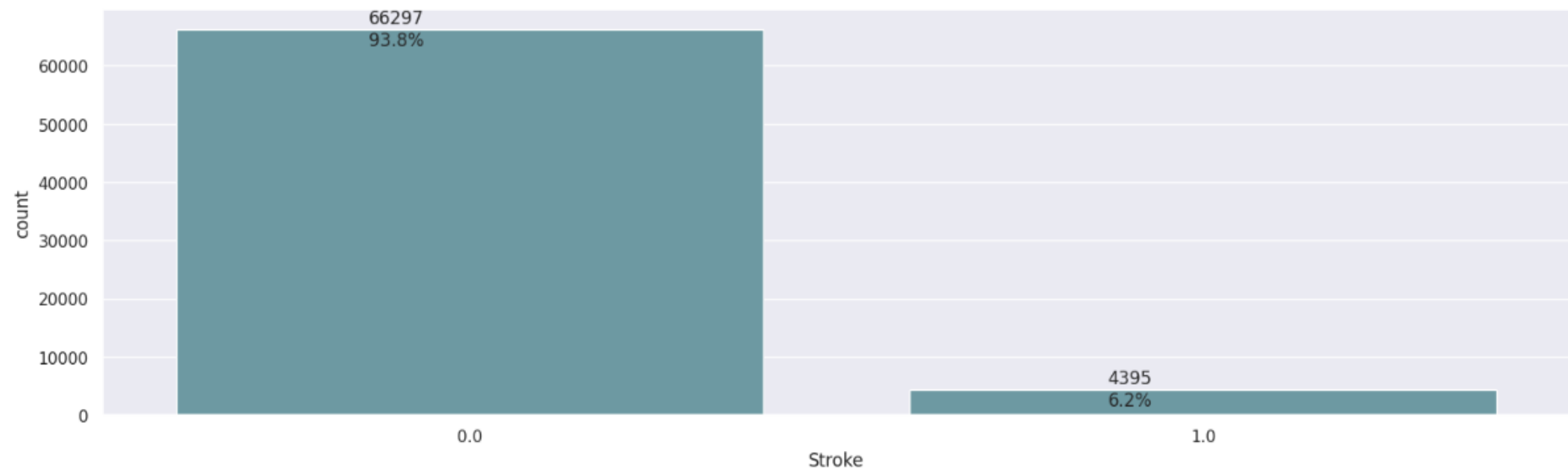
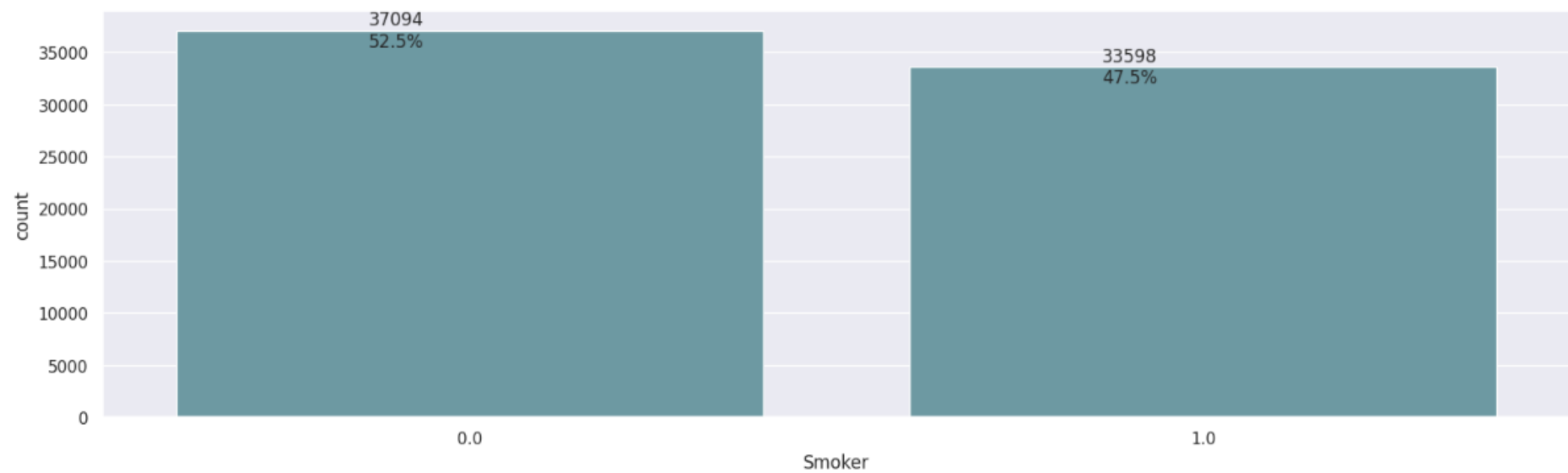
    plt.show()

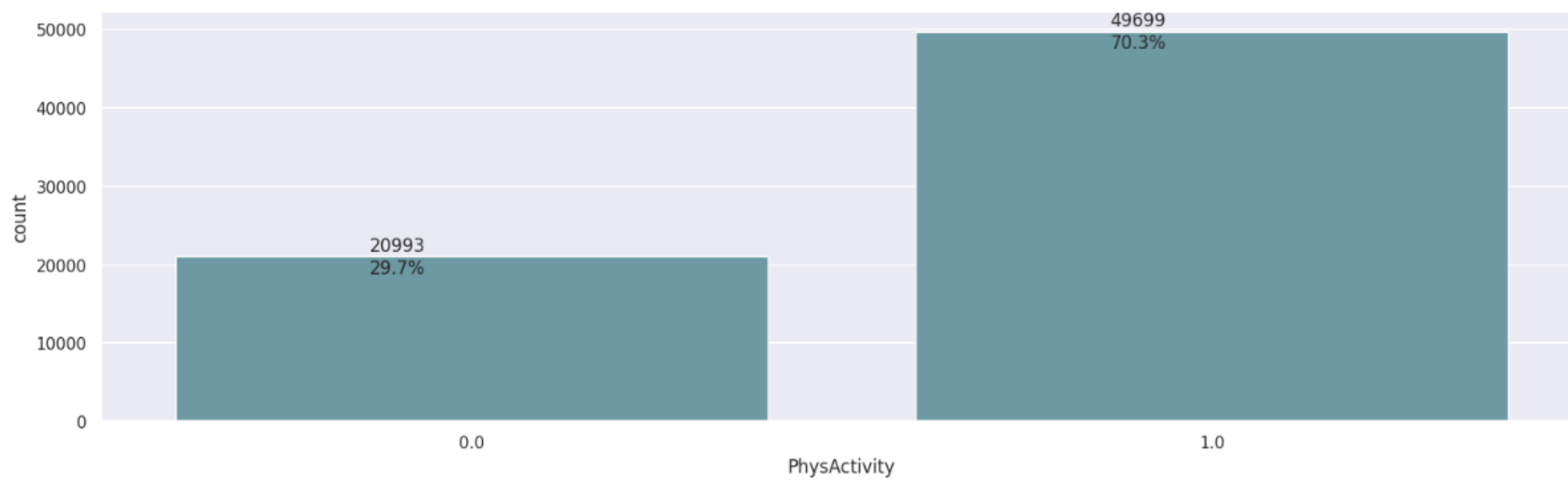
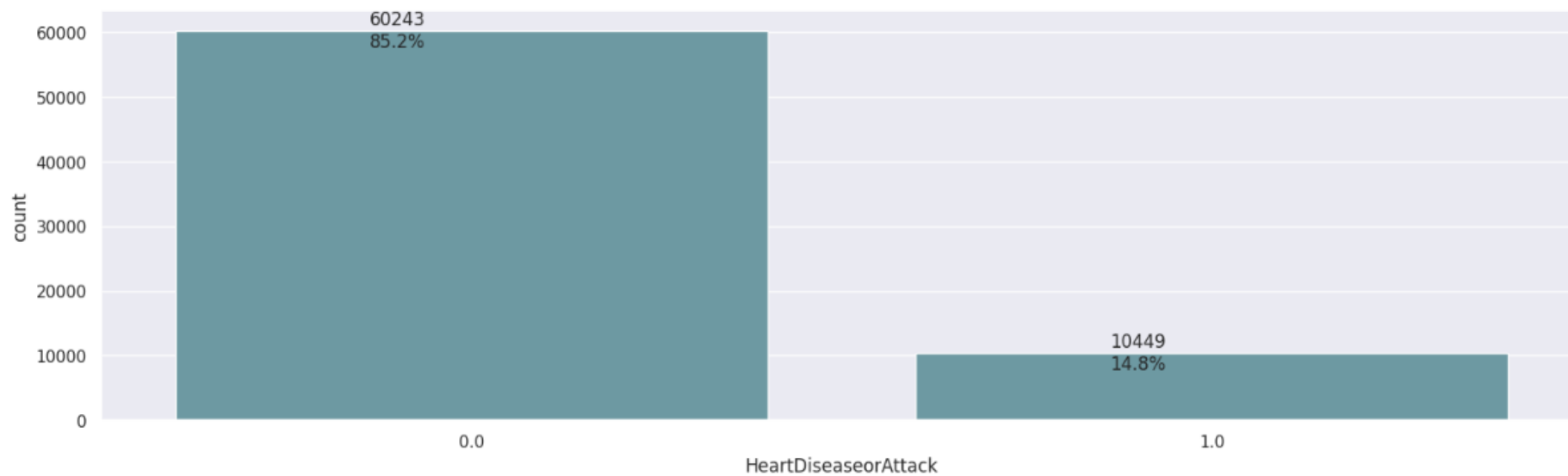
var_cat = ['Diabetes_binary', 'HighBP', 'HighChol', 'CholCheck', 'Smoker', 'Stroke', 'HeartDiseaseorAttack',
           'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth',
           'DiffWalk', 'Sex', 'Age', 'Education', 'Income']

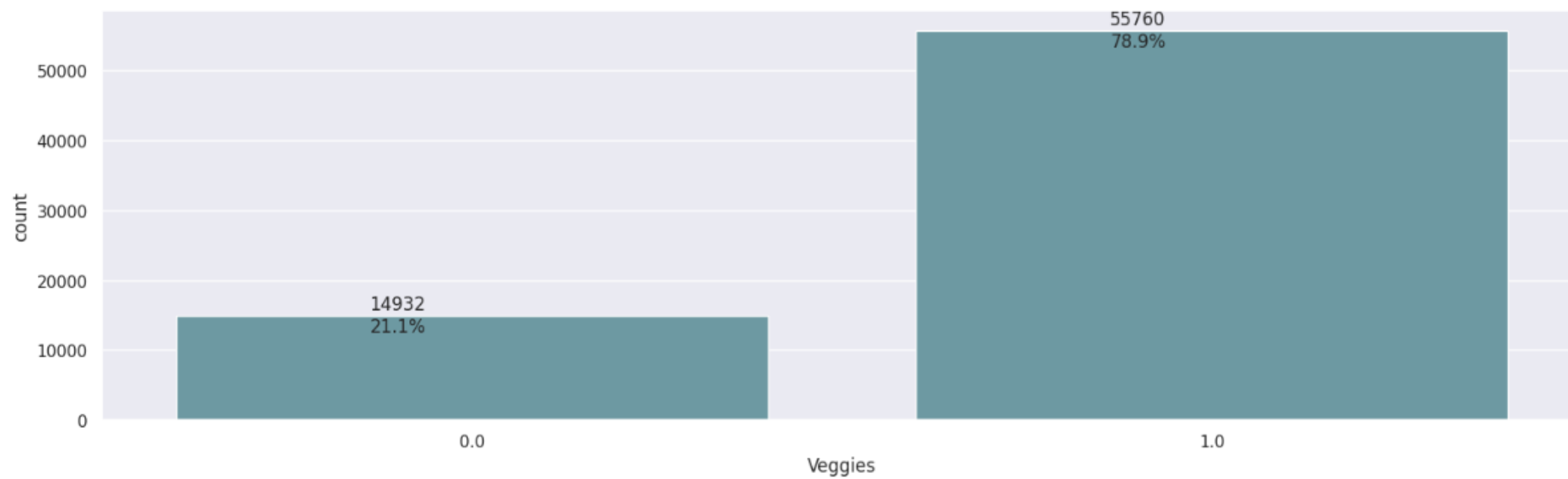
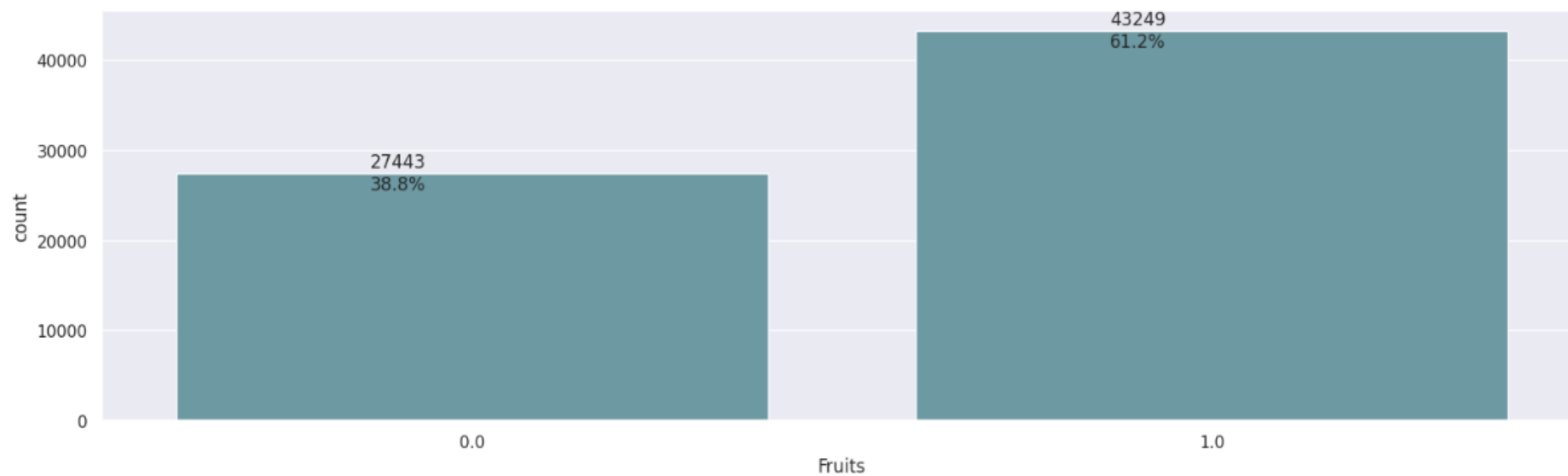
# Generar gráficos para cada variable en var_cat
for variable in var_cat:
    perc_on_bar(variable)
```

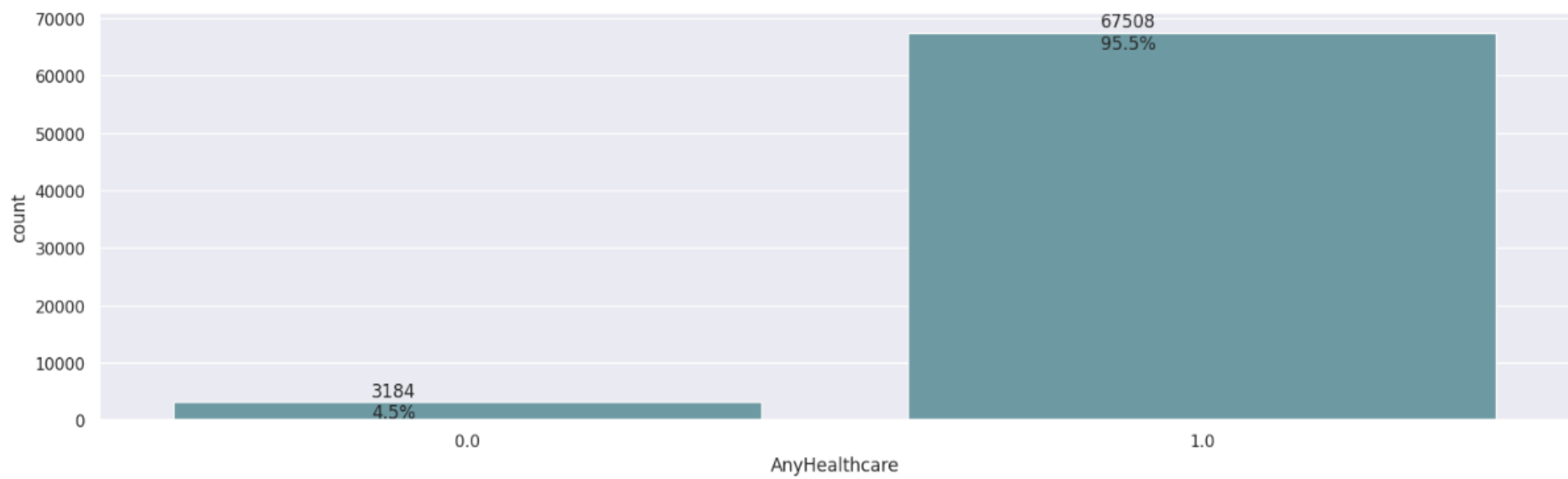
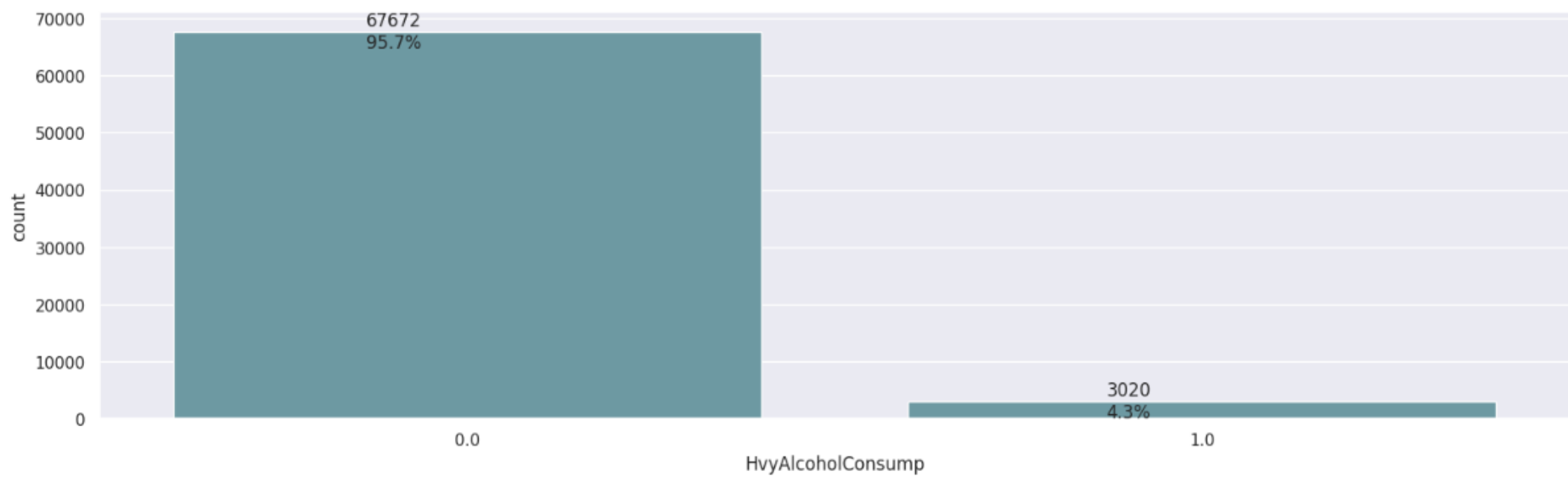


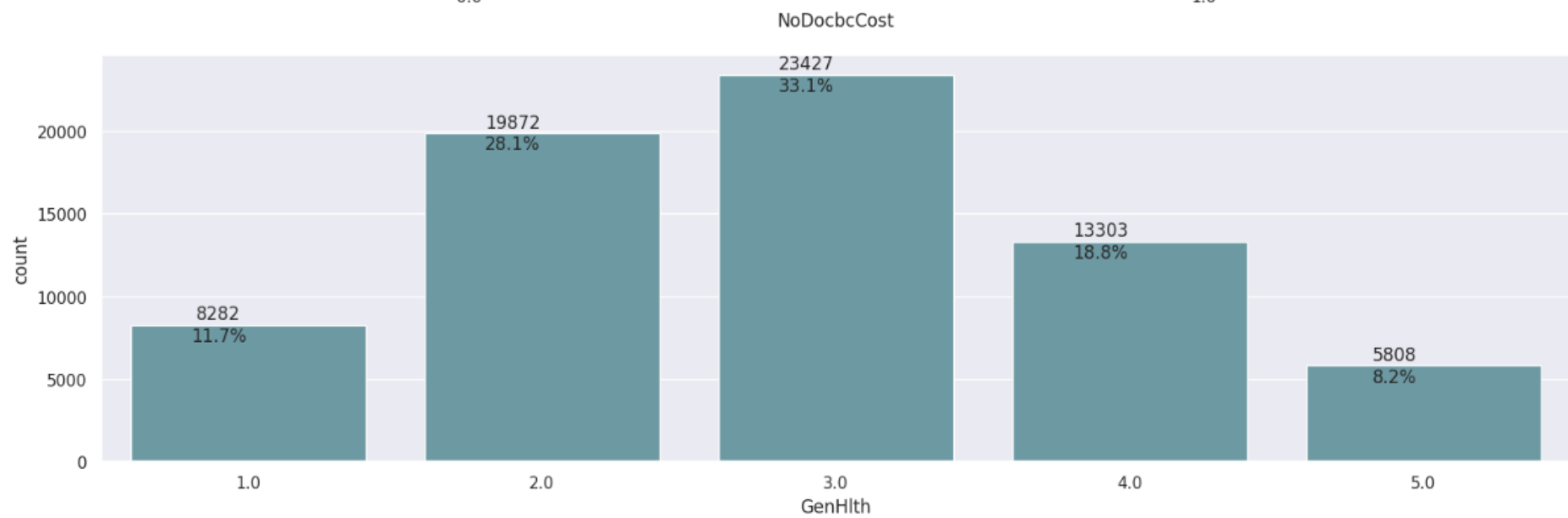
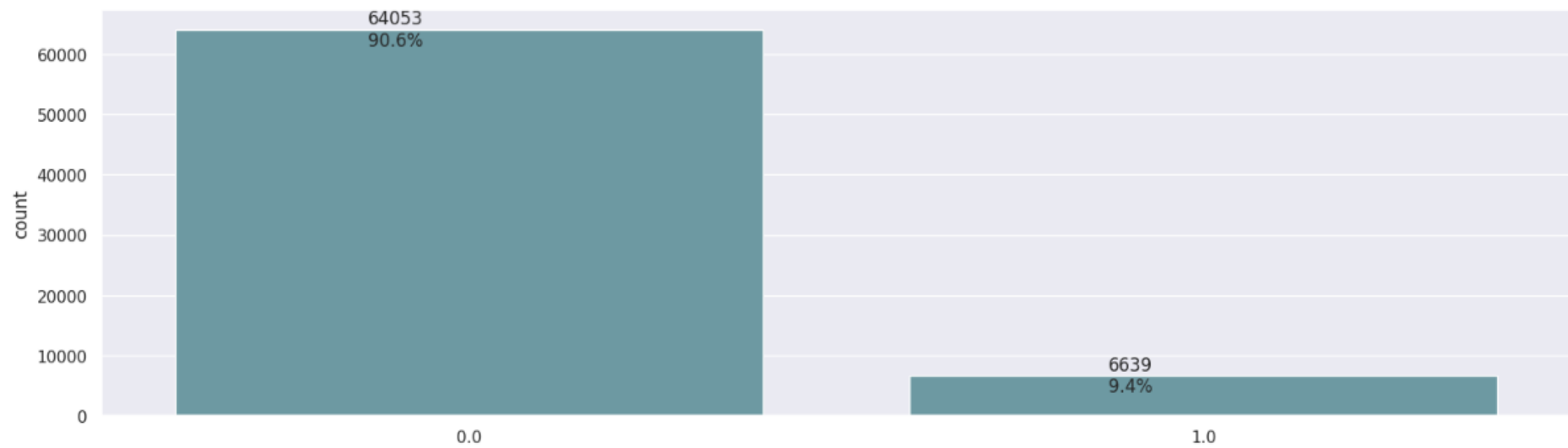


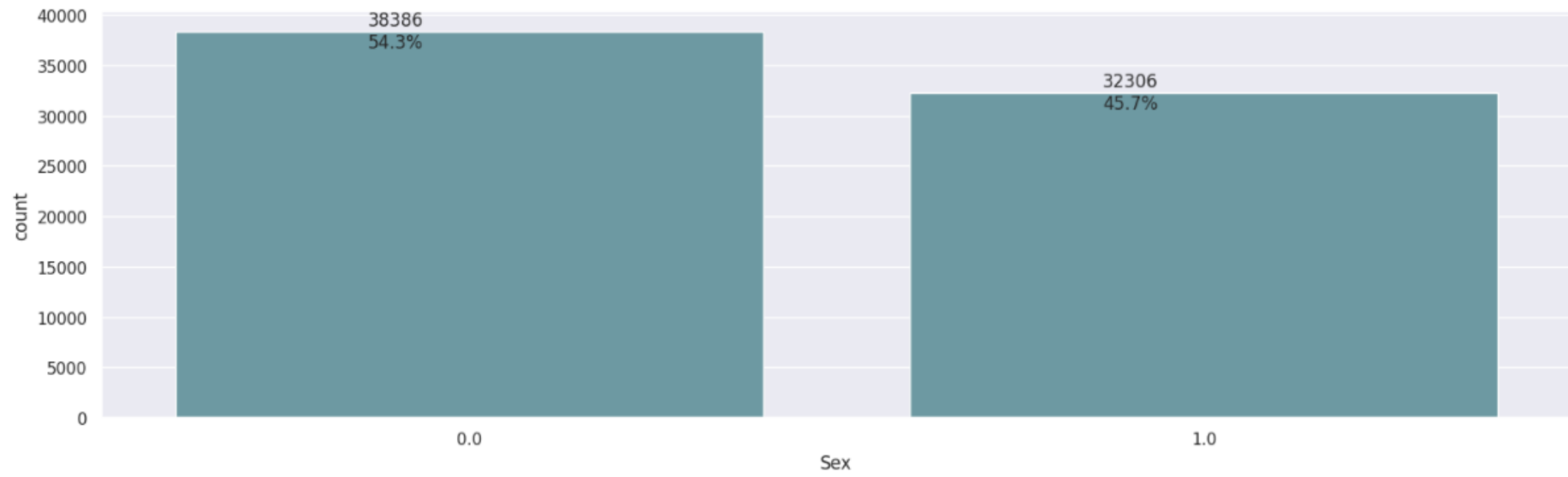
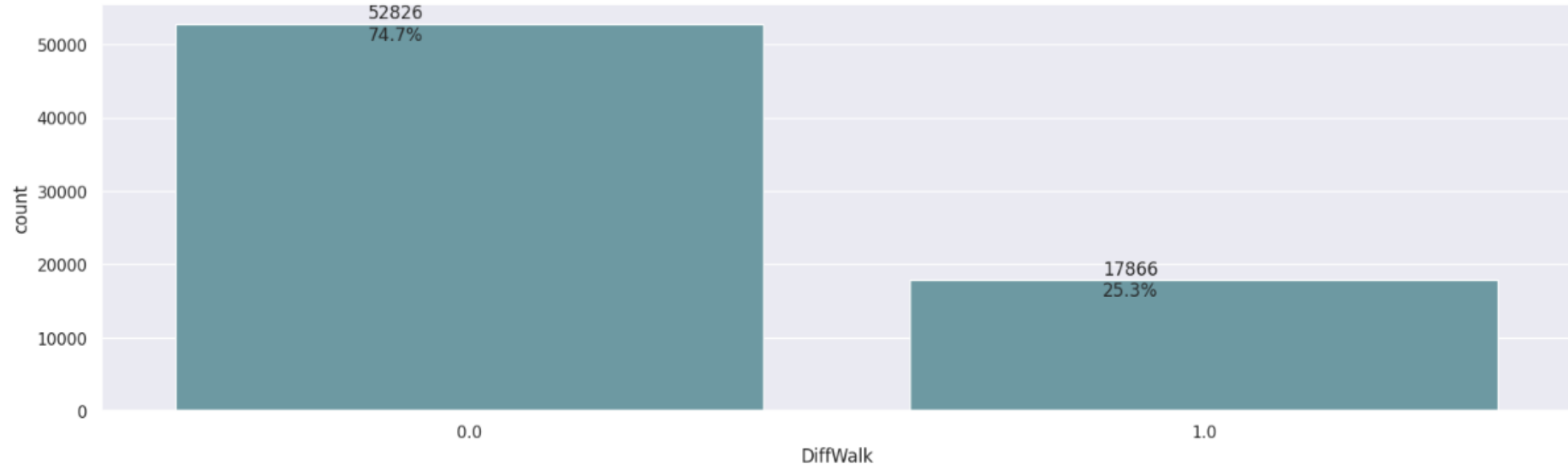


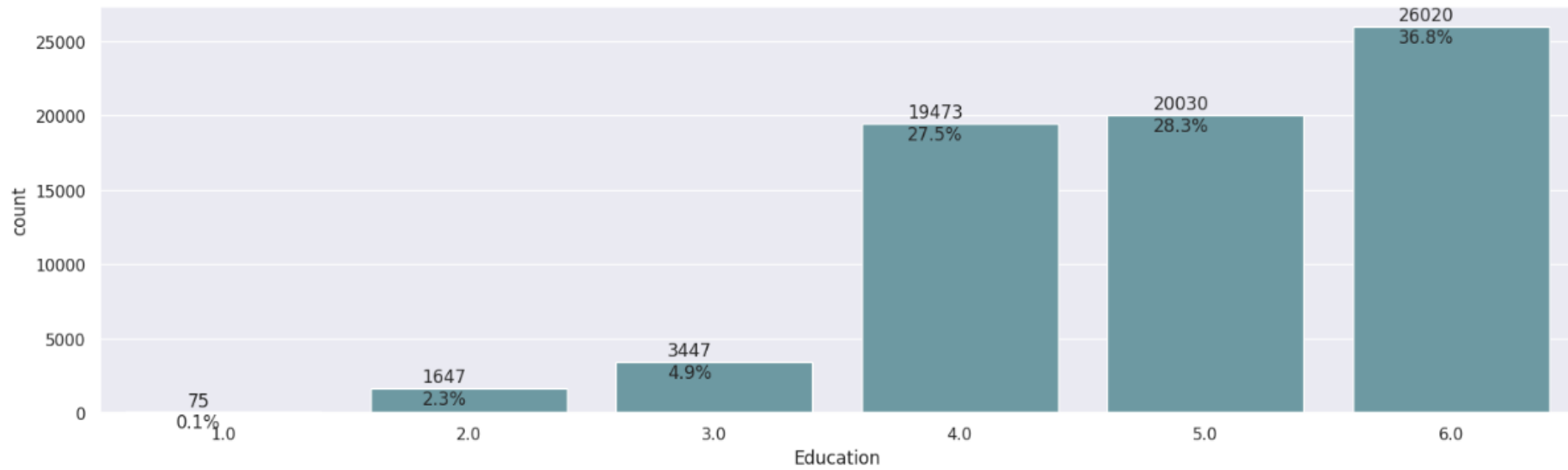
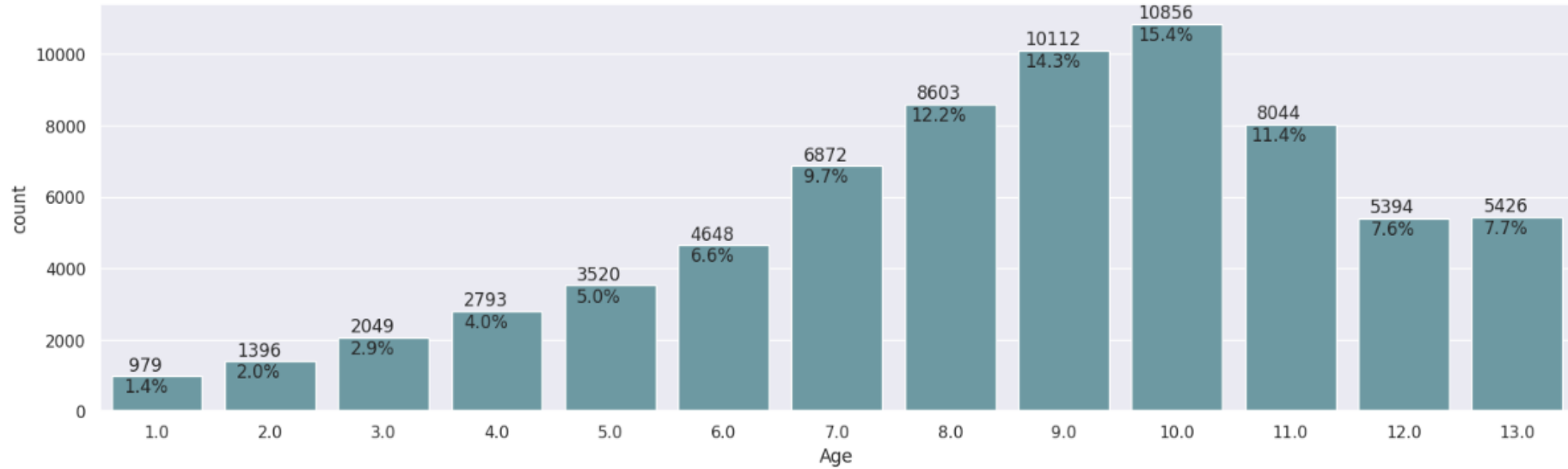


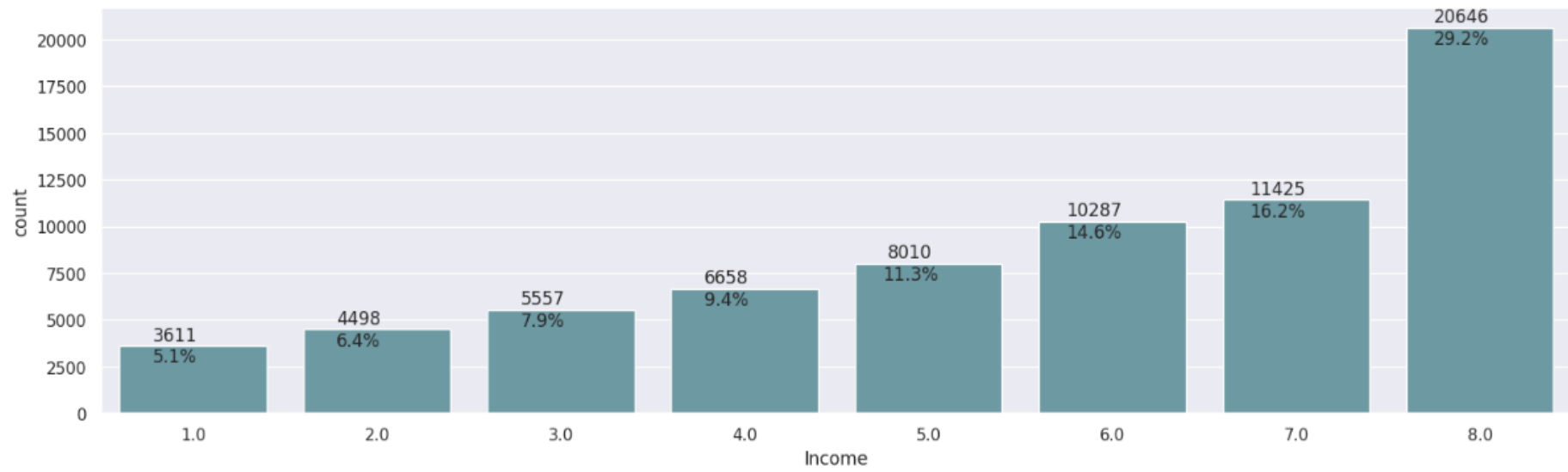












ANÁLISIS BIVARIANTE: VARIABLES CUANTITATIVAS

Procederemos a observar la relación entre nuestra variable objetivo vs las variables cuantitativas para obtener mayores insights.

```
cols1 = ['Diabetes_binary', 'MentHlth', 'PhysHlth', 'BMI']
subset_data1 = data[cols1]
subset_data1.groupby(by='Diabetes_binary').agg('mean')[['MentHlth', 'PhysHlth', 'BMI']]
```

	MentHlth	PhysHlth	BMI
Diabetes_binary			
0.0	3.042268	3.666355	27.769960
1.0	4.461806	7.954479	31.944011

- Las personas que presentan diabetes/prediabetes reportan más días en los que su salud mental no fue buena.
- Las personas que presentan diabetes/prediabetes reportan más días en los que su salud física no fue buena.
- En promedio, las personas que presentan diabetes/prediabetes presentan un BMI más alto.

ANÁLISIS BIVARIANTE: VARIABLES CUALITATIVAS

A continuación, veremos si existe relación entre la variable target y las variables cualitativas.

```
color = '#649EAB'
paleta = sns.color_palette([color, sns.light_palette(color)[2], sns.dark_palette(color)[2]])

def grouped_bar_plot(x, flag=True):
    #sns.set(palette='nipy_spectral')

    #Tabla de frecuencia
    tab1 = pd.crosstab(x, data['Diabetes_binary'], margins=True)

    if flag:
        print(tab1)
        print('-' * 120)

    #Gráfico de barras agrupadas
    ax = tab1.iloc[:-1, :-1].plot(kind='bar', stacked=False, figsize=(12, 6), color=paleta)

    #Añadimos etiquetas y leyenda
    ax.set_ylabel('Frecuencia')
    ax.set_xlabel(x.name)
    plt.title('Diabetes_binary y {}'.format(x.name))
    plt.legend(title='Diabetes_binary')

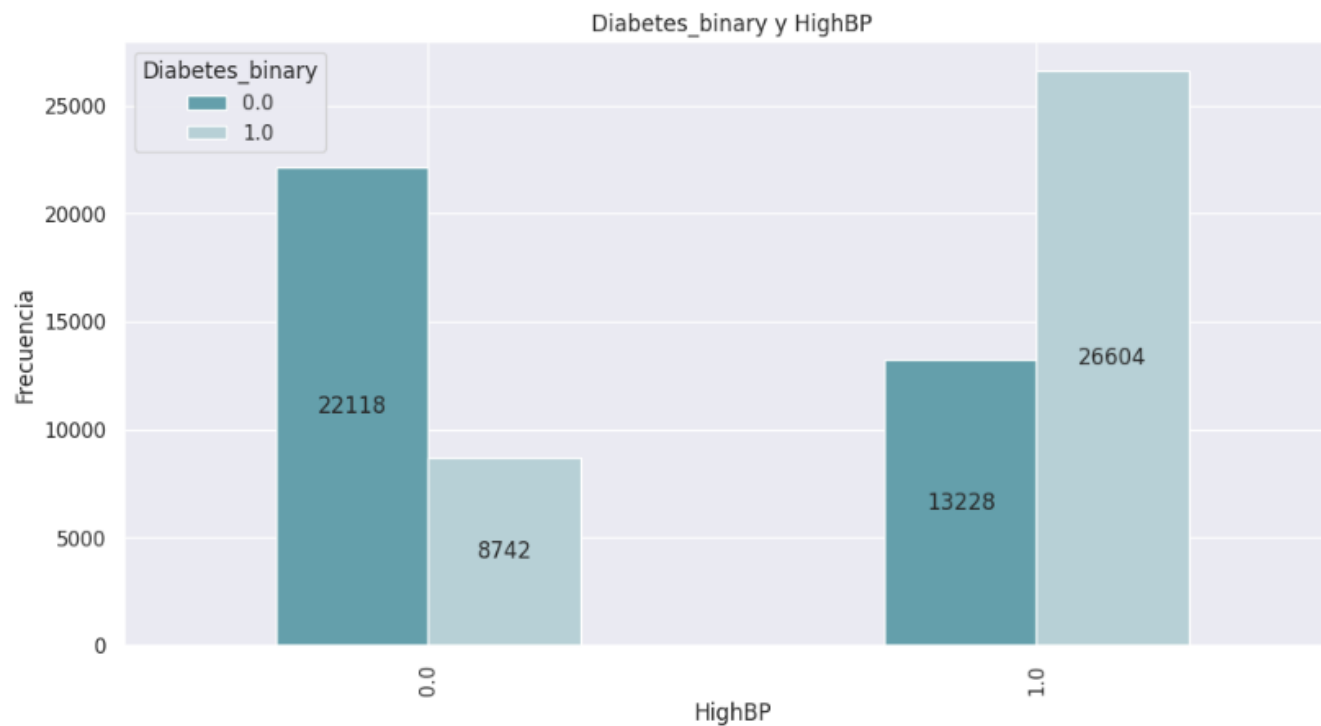
    #Añadimos porcentajes en las barras
    for p in ax.patches:
        width = p.get_width()
        height = p.get_height()
        x, y = p.get_xy()
        ax.annotate(f'{height:}', (x + width / 2, y + height / 2), ha='center', va='center')

    plt.show()
```

Diabetes_binary vs HighBP

```
grouped_bar_plot(data['HighBP'])
```

Diabetes_binary	0.0	1.0	All
HighBP			
0.0	22118	8742	30860
1.0	13228	26604	39832
All	35346	35346	70692



```
import scipy.stats as stats
#Análisis de chi cuadrado para ver si hay una asociación significativa entre ambas variables
crosstab = pd.crosstab(data['Diabetes_binary'], data['HighBP'])
crosstab
stats.chi2_contingency(crosstab)
```

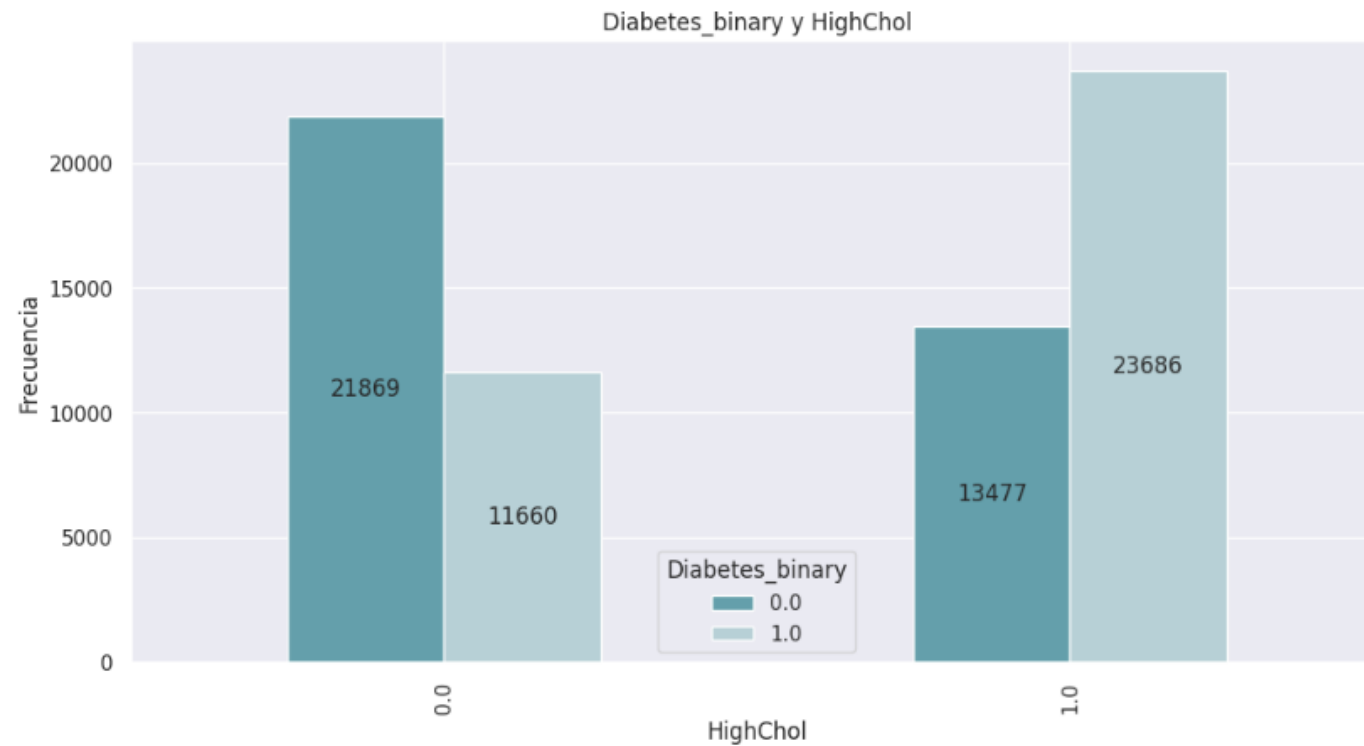
```
Chi2ContingencyResult(statistic=10287.972984997781, pvalue=0.0, dof=1, expected_freq=array([[15430., 19916.],
[15430., 19916.])))
```

➤ Se observa una relación significativa entre ambas variables.

Diabetes_binary vs HighChol

```
grouped_bar_plot(data['HighChol'])
```

Diabetes_binary	0.0	1.0	All
HighChol			
0.0	21869	11660	33529
1.0	13477	23686	37163
All	35346	35346	70692



```
crosstab1 = pd.crosstab(data['Diabetes_binary'], data['HighChol'])
crosstab1
stats.chi2_contingency(crosstab1)
```

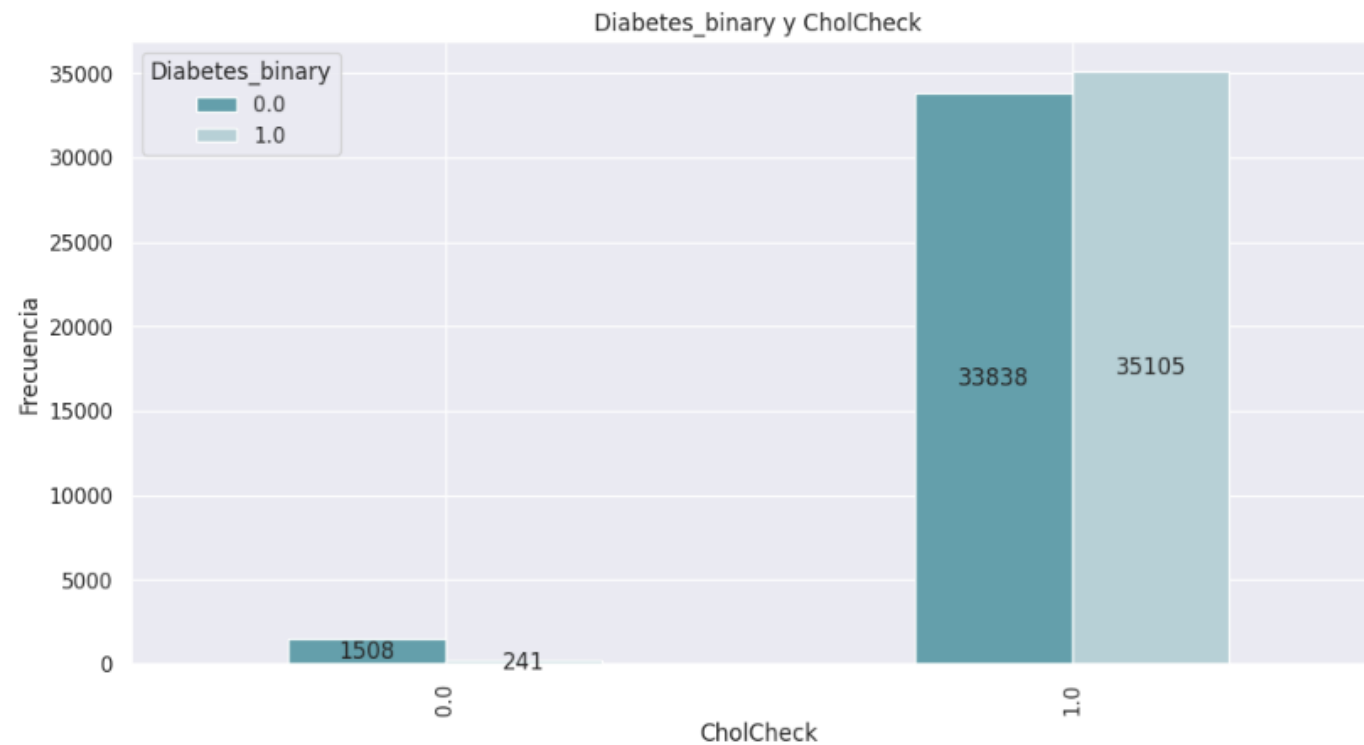
```
Chi2ContingencyResult(statistic=5911.8066998822505, pvalue=0.0, dof=1, expected_freq=array([[16764.5, 18581.5],
[16764.5, 18581.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs CholCheck

```
grouped_bar_plot(data['CholCheck'])
```

Diabetes_binary	0.0	1.0	All
CholCheck			
0.0	1508	241	1749
1.0	33838	35105	68943
All	35346	35346	70692



```
crosstab2 = pd.crosstab(data['Diabetes_binary'], data['CholCheck'])
crosstab2
stats.chi2_contingency(crosstab2)
```

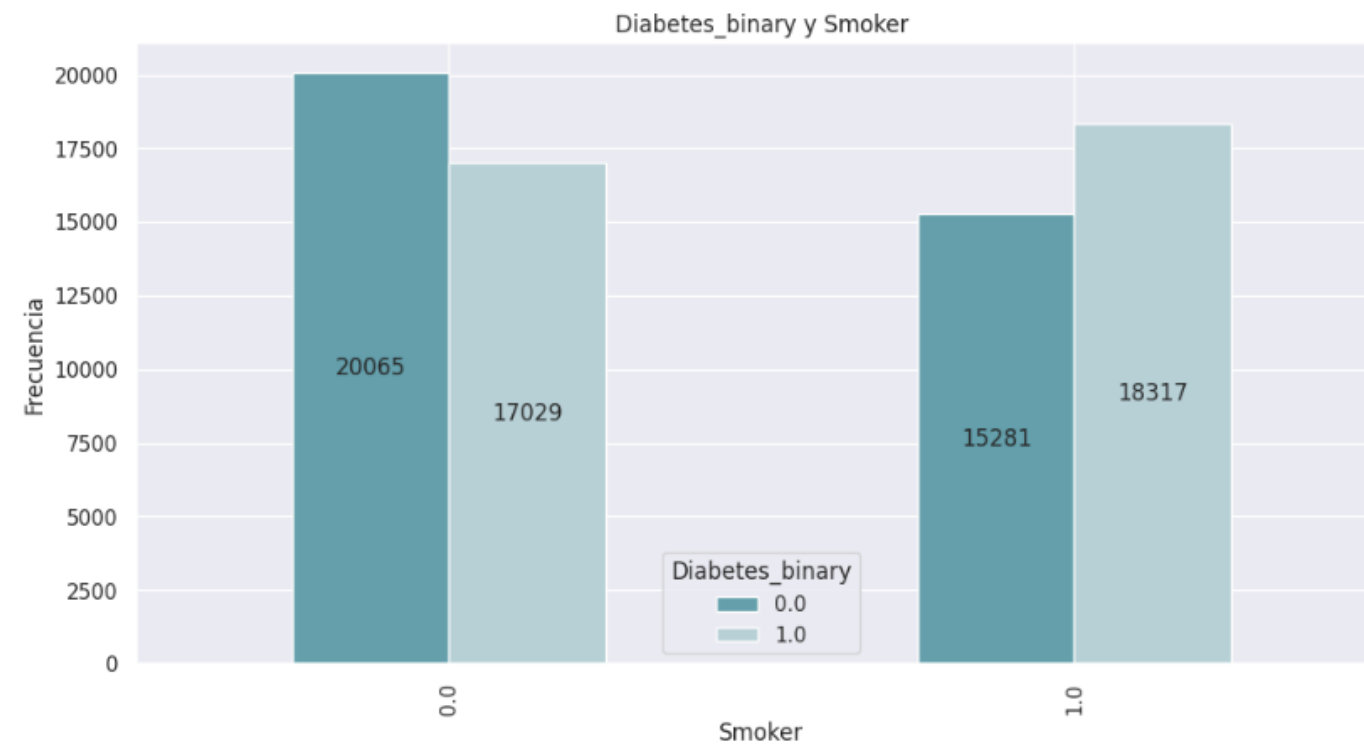
```
Chi2ContingencyResult(statistic=939.6317718798653, pvalue=2.3798712773419562e-206, dof=1, expected_freq=array([[ 874.5, 34471.5],
[ 874.5, 34471.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Smoker

```
grouped_bar_plot(data['Smoker'])
```

Diabetes_binary	0.0	1.0	All
Smoker			
0.0	20065	17029	37094
1.0	15281	18317	33598
All	35346	35346	70692



```
crosstab3 = pd.crosstab(data['Diabetes_binary'], data['Smoker'])
```

```
crosstab3
```

```
stats.chi2_contingency(crosstab3)
```

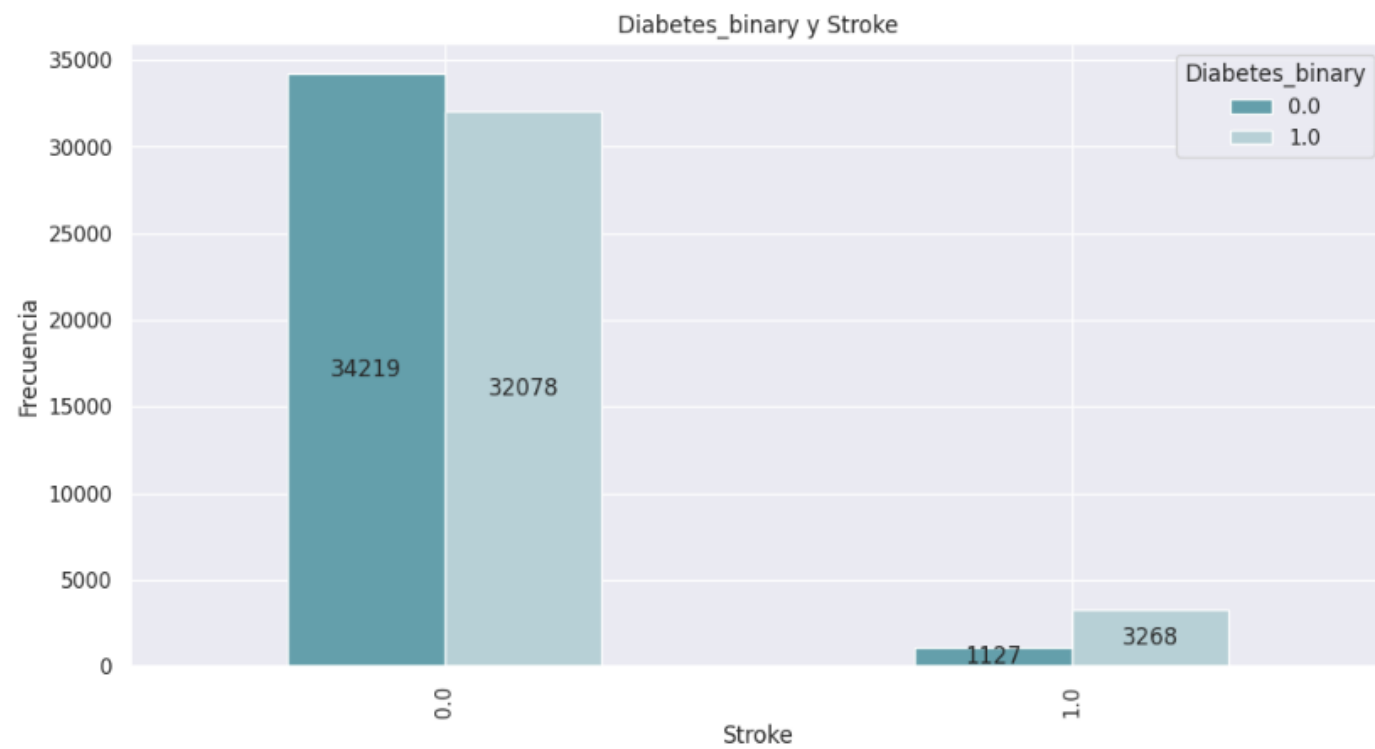
```
Chi2ContingencyResult(statistic=522.4810772937883, pvalue=1.2211053080400827e-115, dof=1, expected_freq=array([[18547., 16799.],
[18547., 16799.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Stroke

```
grouped_bar_plot(data['Stroke'])
```

Diabetes_binary	0.0	1.0	All
Stroke			
0.0	34219	32078	66297
1.0	1127	3268	4395
All	35346	35346	70692



```
crosstab4 = pd.crosstab(data['Diabetes_binary'], data['Stroke'])
crosstab4
stats.chi2_contingency(crosstab4)
```

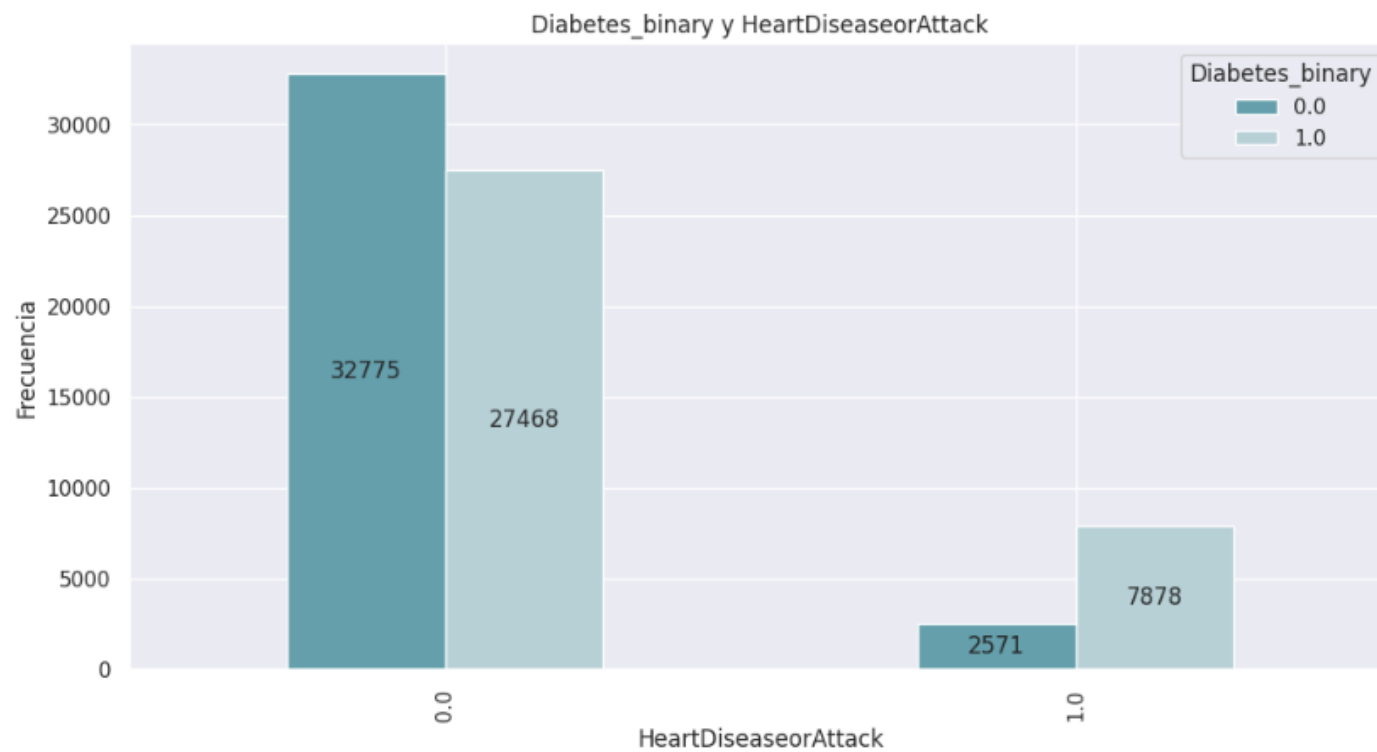
```
Chi2ContingencyResult(statistic=1111.0793074560897, pvalue=1.290837098922016e-243, dof=1, expected_freq=array([[33148.5, 2197.5],
[33148.5, 2197.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs HeartDiseaseorAttack

```
grouped_bar_plot(data['HeartDiseaseorAttack'])
```

Diabetes_binary	0.0	1.0	All
HeartDiseaseorAttack			
0.0	32775	27468	60243
1.0	2571	7878	10449
All	35346	35346	70692



```
crosstab5 = pd.crosstab(data['Diabetes_binary'], data['HeartDiseaseorAttack'])
crosstab5
stats.chi2_contingency(crosstab5)
```

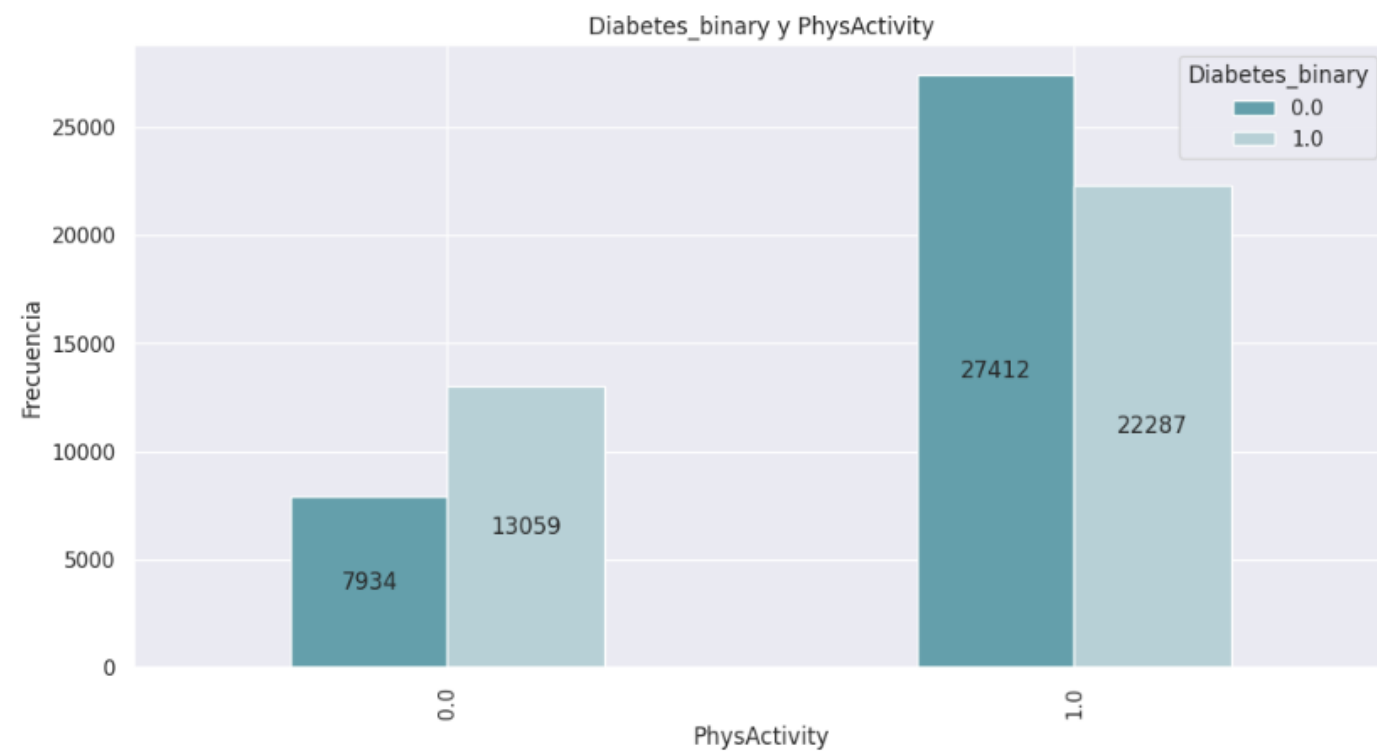
```
Chi2ContingencyResult(statistic=3161.7202445322782, pvalue=0.0, dof=1, expected_freq=array([[30121.5, 5224.5],
[30121.5, 5224.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs PhysActivity

```
grouped_bar_plot(data['PhysActivity'])
```

Diabetes_binary	0.0	1.0	All
PhysActivity			
0.0	7934	13059	20993
1.0	27412	22287	49699
All	35346	35346	70692



```
crosstab6 = pd.crosstab(data['Diabetes_binary'], data['PhysActivity'])
crosstab6
stats.chi2_contingency(crosstab6)
```

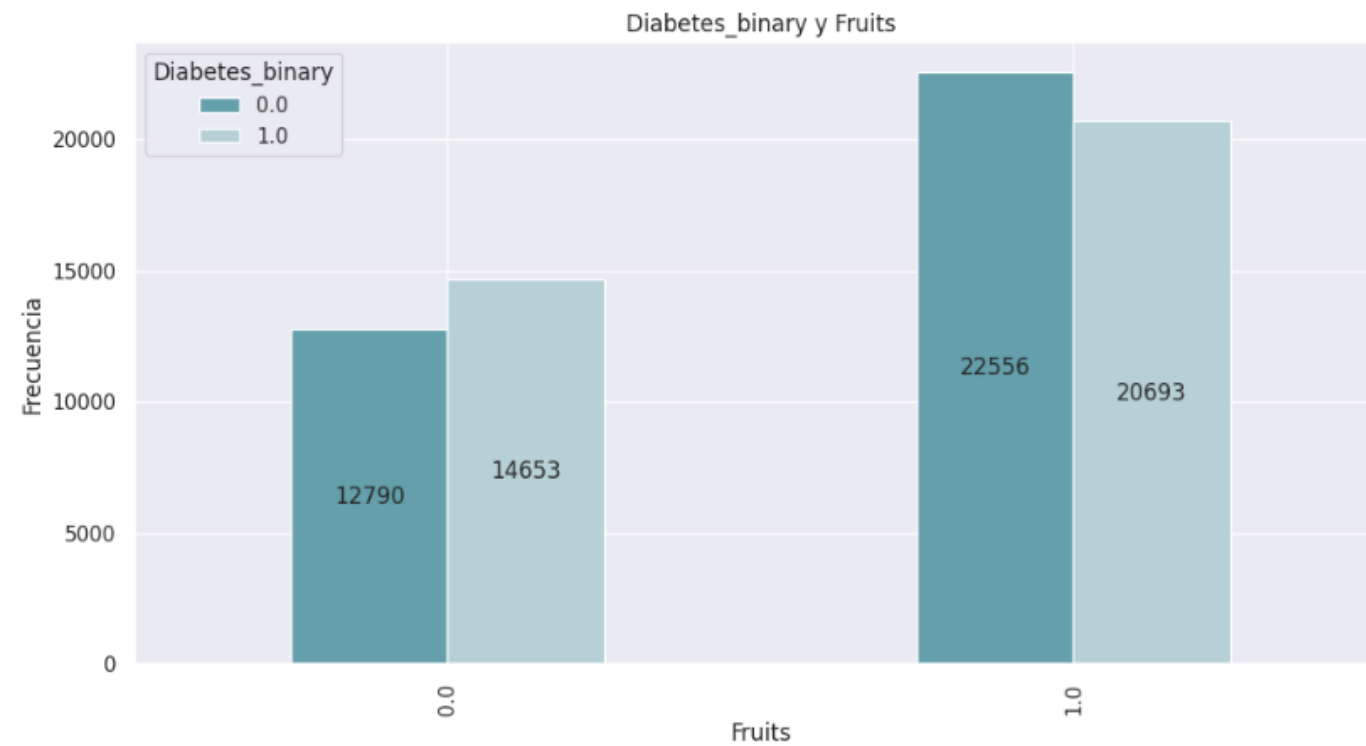
```
Chi2ContingencyResult(statistic=1778.9607035956992, pvalue=0.0, dof=1, expected_freq=array([[10496.5, 24849.5],
[10496.5, 24849.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Fruits

```
grouped_bar_plot(data['Fruits'])
```

Diabetes_binary	0.0	1.0	All
Fruits			
0.0	12790	14653	27443
1.0	22556	20693	43249
All	35346	35346	70692



```
crosstab7 = pd.crosstab(data['Diabetes_binary'], data['Fruits'])
crosstab7
stats.chi2_contingency(crosstab7)
```

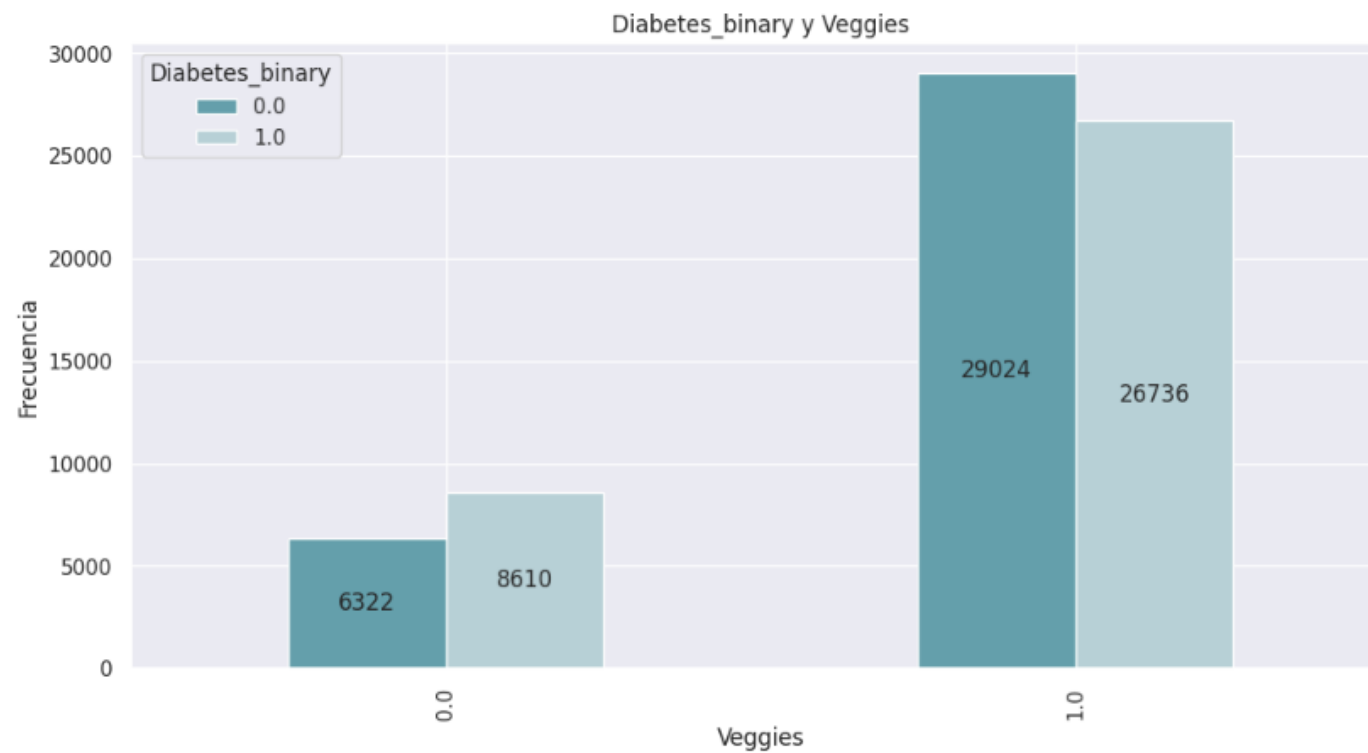
```
Chi2ContingencyResult(statistic=206.50090830615105, pvalue=7.967064756507964e-47, dof=1, expected_freq=array([[13721.5, 21624.5],
[13721.5, 21624.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Veggies

```
grouped_bar_plot(data['Veggies'])
```

Diabetes_binary	0.0	1.0	All
Veggies			
0.0	6322	8610	14932
1.0	29024	26736	55760
All	35346	35346	70692



```
crosstab8 = pd.crosstab(data['Diabetes_binary'], data['Veggies'])
crosstab8
stats.chi2_contingency(crosstab8)
```

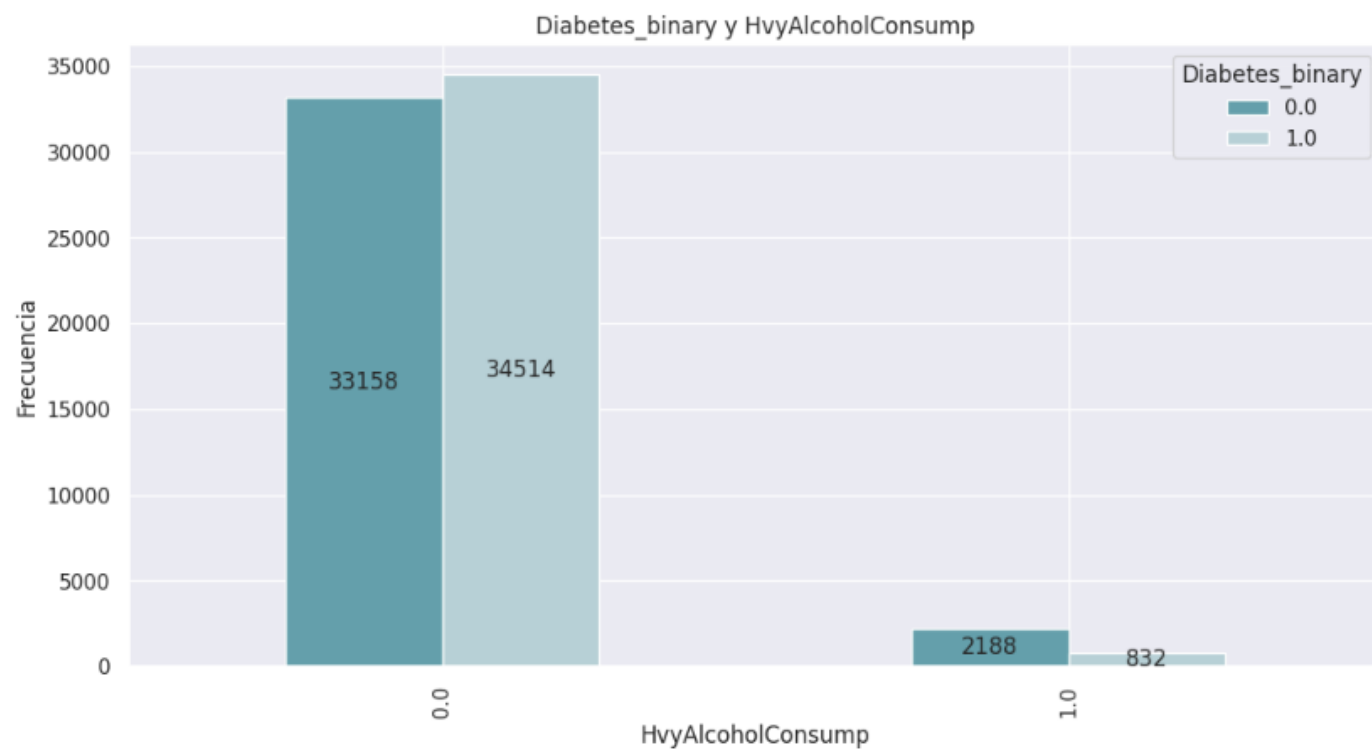
```
Chi2ContingencyResult(statistic=444.0806516898606, pvalue=1.4007103685991128e-98, dof=1, expected_freq=array([[ 7466., 27880.],
[ 7466., 27880.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs HvyAlcoholConsump

```
grouped_bar_plot(data['HvyAlcoholConsump'])
```

Diabetes_binary	0.0	1.0	All
HvyAlcoholConsump			
0.0	33158	34514	67672
1.0	2188	832	3020
All	35346	35346	70692



```
crosstab9 = pd.crosstab(data['Diabetes_binary'], data['HvyAlcoholConsump'])
crosstab9
stats.chi2_contingency(crosstab9)
```

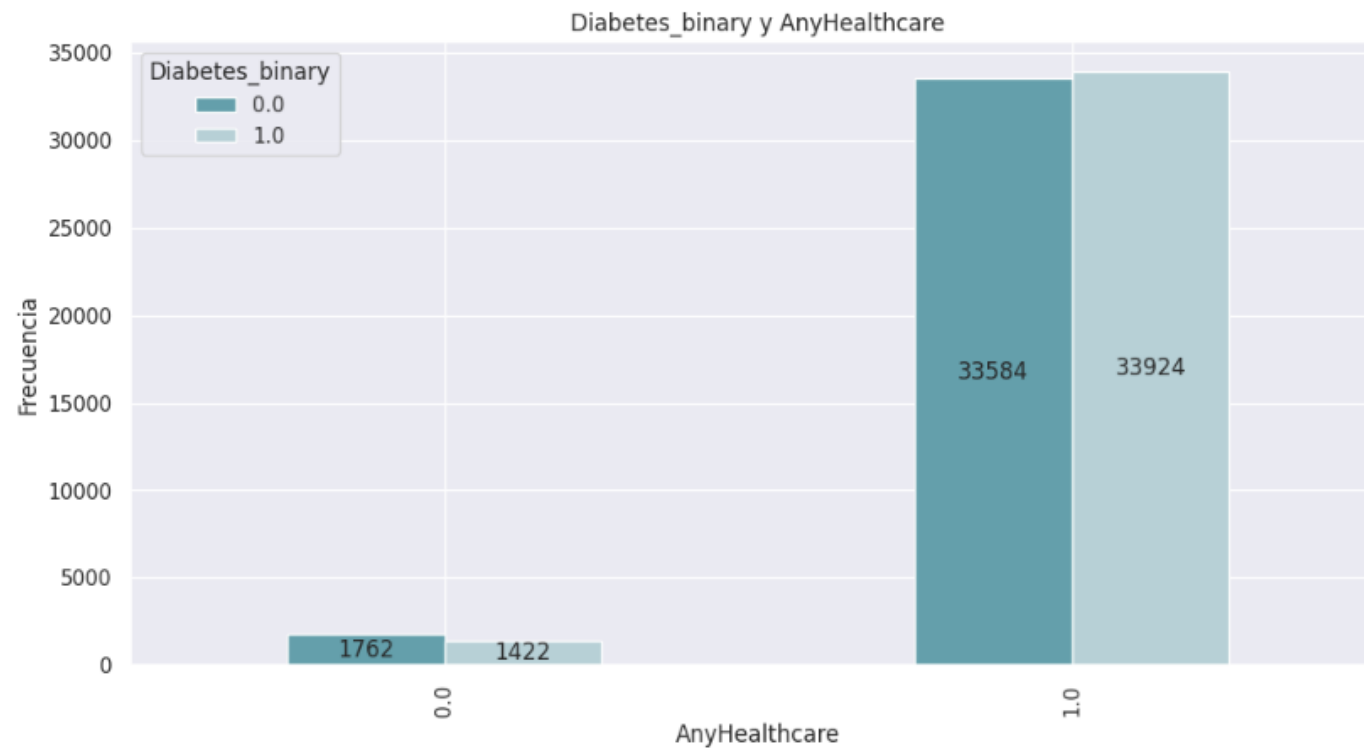
```
Chi2ContingencyResult(statistic=635.0865339749427, pvalue=3.9133962745676324e-140, dof=1, expected_freq=array([[33836., 1510.],
[33836., 1510.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs AnyHealthcare

```
grouped_bar_plot(data['AnyHealthcare'])
```

Diabetes_binary	0.0	1.0	All
AnyHealthcare			
0.0	1762	1422	3184
1.0	33584	33924	67508
All	35346	35346	70692



```
crosstab10 = pd.crosstab(data['Diabetes_binary'], data['AnyHealthcare'])
crosstab10
stats.chi2_contingency(crosstab10)
```

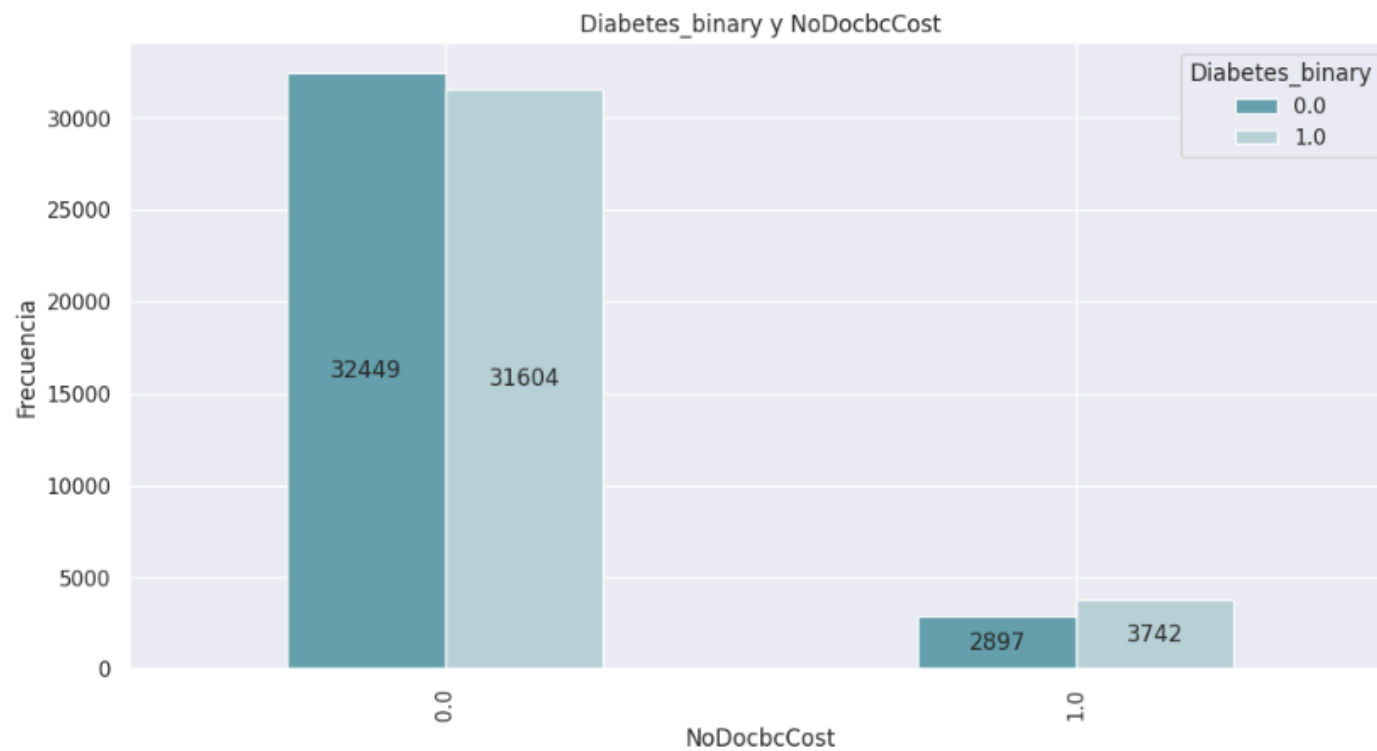
```
Chi2ContingencyResult(statistic=37.79561046998934, pvalue=7.855833890083924e-10, dof=1, expected_freq=array([[ 1592., 33754.],
[ 1592., 33754.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs NoDocbcCost

```
grouped_bar_plot(data['NoDocbcCost'])
```

Diabetes_binary	0.0	1.0	All
NoDocbcCost			
0.0	32449	31604	64053
1.0	2897	3742	6639
All	35346	35346	70692



```
crosstab11 = pd.crosstab(data['Diabetes_binary'], data['NoDocbcCost'])
crosstab11
stats.chi2_contingency(crosstab11)
```

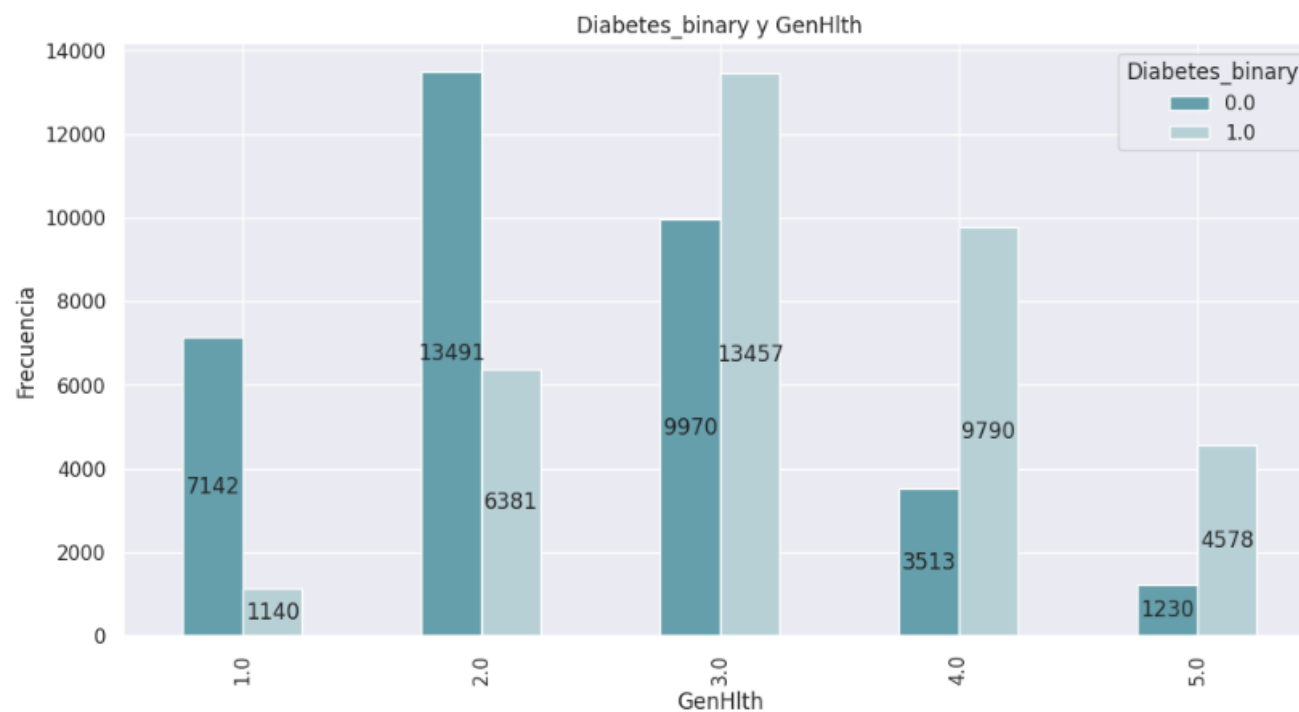
```
Chi2ContingencyResult(statistic=118.4167174482265, pvalue=1.4053255735064045e-27, dof=1, expected_freq=array([[32026.5, 3319.5],
[32026.5, 3319.5]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs GenHlth

```
grouped_bar_plot(data['GenHlth'])
```

Diabetes_binary	0.0	1.0	All
GenHlth			
1.0	7142	1140	8282
2.0	13491	6381	19872
3.0	9970	13457	23427
4.0	3513	9790	13303
5.0	1230	4578	5808
All	35346	35346	70692



```
crosstab12 = pd.crosstab(data['Diabetes_binary'], data['GenHlth'])
crosstab12
stats.chi2_contingency(crosstab12)
```

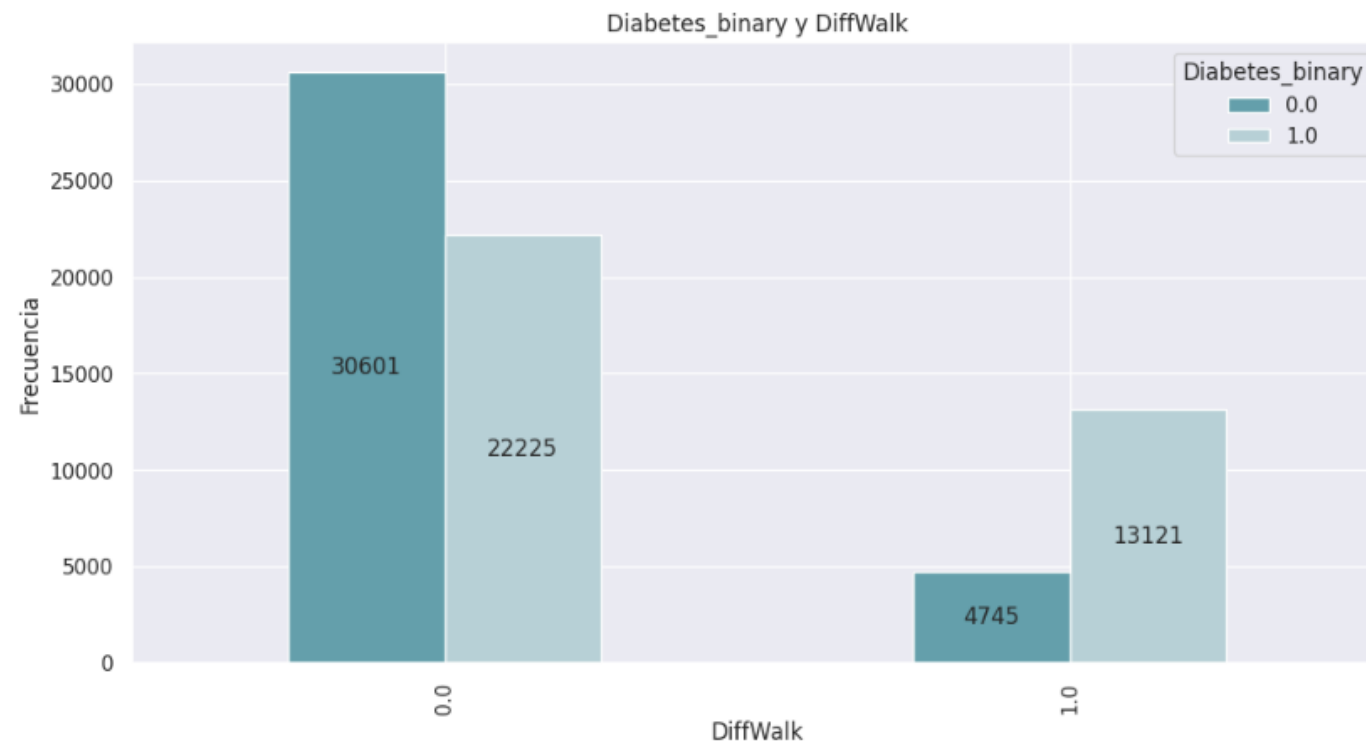
```
Chi2ContingencyResult(statistic=12304.318979903528, pvalue=0.0, dof=4, expected_freq=array([[ 4141. ,  9936. , 11713.5,  6651.5,  2904. ],
[ 4141. ,  9936. , 11713.5,  6651.5,  2904. ]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs DiffWalk

```
grouped_bar_plot(data['DiffWalk'])
```

Diabetes_binary	0.0	1.0	All
DiffWalk			
0.0	30601	22225	52826
1.0	4745	13121	17866
All	35346	35346	70692



```
crosstab13 = pd.crosstab(data['Diabetes_binary'], data['DiffWalk'])
crosstab13
stats.chi2_contingency(crosstab13)
```

```
Chi2ContingencyResult(statistic=5253.694843161374, pvalue=0.0, dof=1, expected_freq=array([[26413., 8933.],
[26413., 8933.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Sex

```
grouped_bar_plot(data['Sex'])
```

Diabetes_binary	0.0	1.0	All
Sex			
0.0	19975	18411	38386
1.0	15371	16935	32306
All	35346	35346	70692



```
crosstab14 = pd.crosstab(data['Diabetes_binary'], data['Sex'])
crosstab14
stats.chi2_contingency(crosstab14)
```

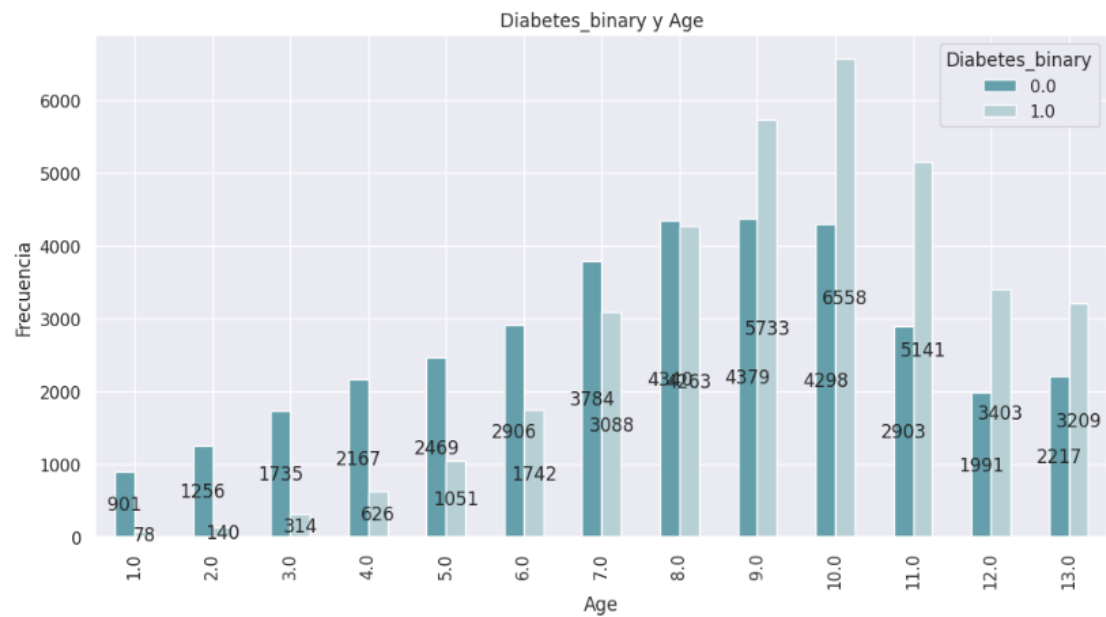
```
Chi2ContingencyResult(statistic=139.26185542886512, pvalue=3.860395909809483e-32, dof=1, expected_freq=array([[19193., 16153.],
[19193., 16153.])))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Age

```
grouped_bar_plot(data['Age'])
```

Diabetes_binary	0.0	1.0	All
Age			
1.0	901	78	979
2.0	1256	140	1396
3.0	1735	314	2049
4.0	2167	626	2793
5.0	2469	1051	3520
6.0	2906	1742	4648
7.0	3784	3088	6872
8.0	4340	4263	8603
9.0	4379	5733	10112
10.0	4298	6558	10856
11.0	2903	5141	8044
12.0	1991	3403	5394
13.0	2217	3209	5426
All	35346	35346	70692



```
crosstab15 = pd.crosstab(data['Diabetes_binary'], data['Age'])
crosstab15
stats.chi2_contingency(crosstab15)
```

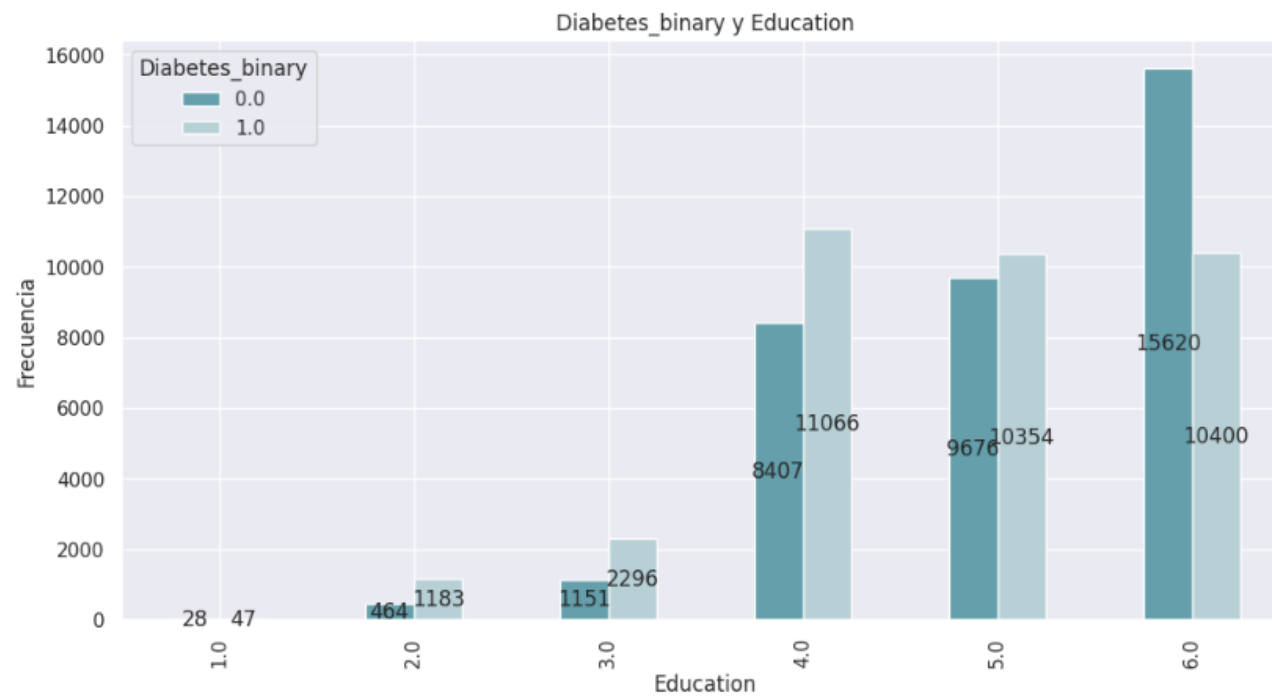
```
Chi2ContingencyResult(statistic=6179.057132257292, pvalue=0.0, dof=12, expected_freq=array([[ 489.5,  698. , 1024.5, 1396.5, 1760. , 2324. , 3436. , 4301.5,
  5056. , 5428. , 4022. , 2697. , 2713. ],
[ 489.5,  698. , 1024.5, 1396.5, 1760. , 2324. , 3436. , 4301.5,
  5056. , 5428. , 4022. , 2697. , 2713. ]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Education

```
grouped_bar_plot(data['Education'])
```

Diabetes_binary	0.0	1.0	All
Education			
1.0	28	47	75
2.0	464	1183	1647
3.0	1151	2296	3447
4.0	8407	11066	19473
5.0	9676	10354	20030
6.0	15620	10400	26020
All	35346	35346	70692



```
crosstab16 = pd.crosstab(data['Diabetes_binary'], data['Education'])
crosstab16
stats.chi2_contingency(crosstab16)
```

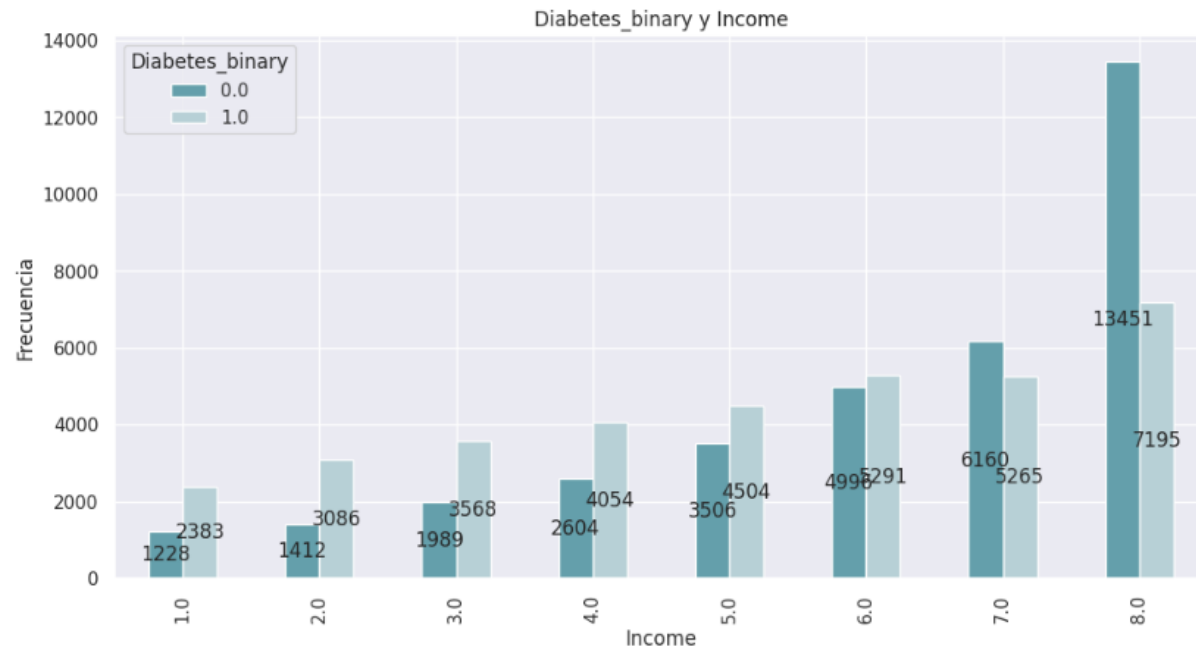
```
Chi2ContingencyResult(statistic=2132.272551584347, pvalue=0.0, dof=5, expected_freq=array([[ 37.5,  823.5, 1723.5, 9736.5, 10015. , 13010. ],
[ 37.5,  823.5, 1723.5, 9736.5, 10015. , 13010. ]]))
```

► Se observa una relación significativa entre ambas variables.

Diabetes_binary vs Income

```
grouped_bar_plot(data['Income'])
```

Diabetes_binary	0.0	1.0	All
Income			
1.0	1228	2383	3611
2.0	1412	3086	4498
3.0	1989	3568	5557
4.0	2604	4054	6658
5.0	3506	4504	8010
6.0	4996	5291	10287
7.0	6160	5265	11425
8.0	13451	7195	20646
All	35346	35346	70692



```
crosstab17 = pd.crosstab(data['Diabetes_binary'], data['Income'])
crosstab17
stats.chi2_contingency(crosstab17)
```

```
Chi2ContingencyResult(statistic=3855.454575562344, pvalue=0.0, dof=7, expected_freq=array([[ 1805.5,  2249. ,  2778.5,  3329. ,  4005. ,  5143.5,  5712.5,
 10323. ],
[ 1805.5,  2249. ,  2778.5,  3329. ,  4005. ,  5143.5,  5712.5,
 10323. ]]))
```

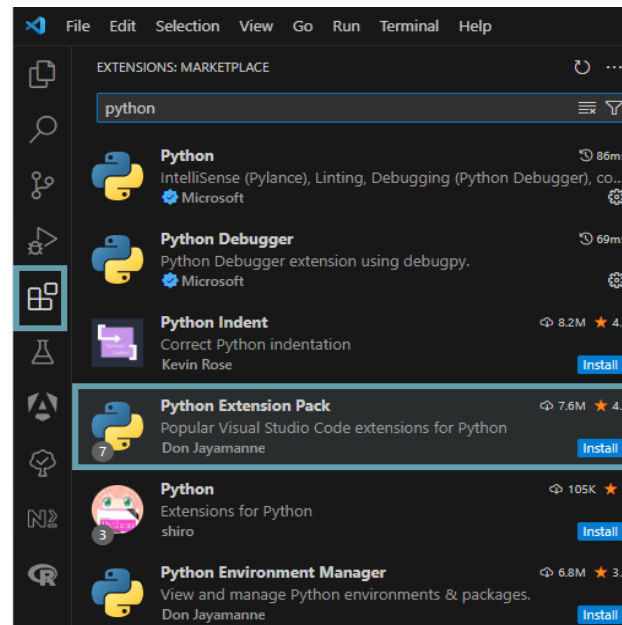
► Se observa una relación significativa entre ambas variables.

PASO A PASO: CREACIÓN DE SERVICIO WEB USANDO FASTAPI

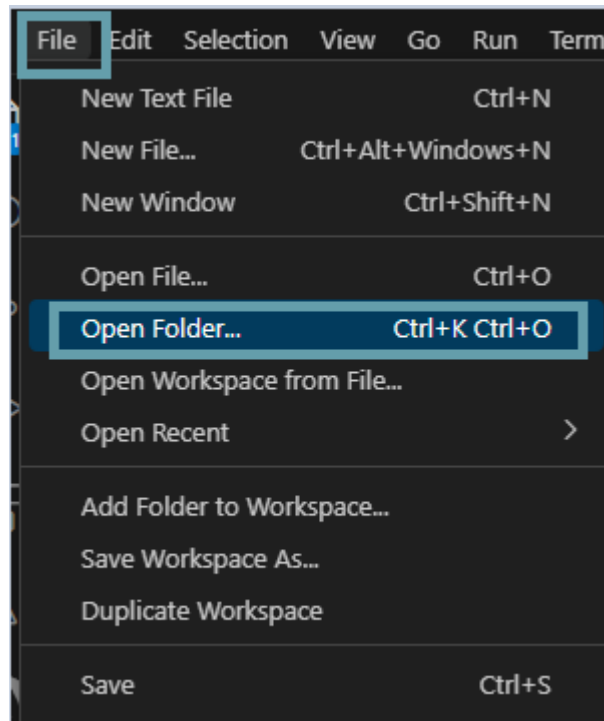
Herramientas:

IDE	Lenguaje	Librerías
Visual Studio Code	Python	FastAPI uvicorn CORSMiddleware pickle pandas scikit-learn

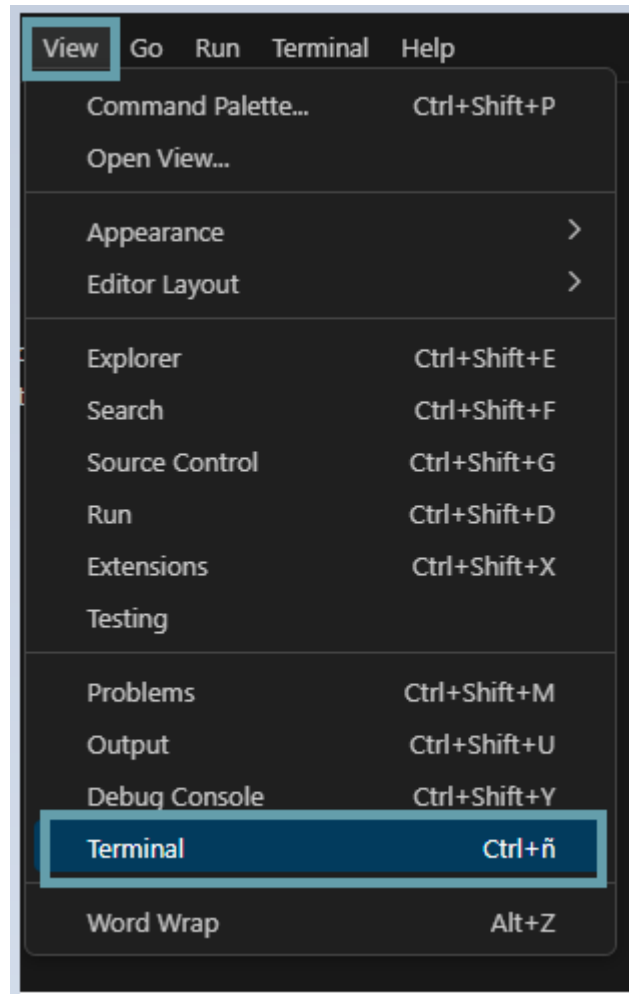
1. Verificar si contamos con la extensión de Python:



2. Abrir el folder en donde tenemos el archivo pickle y en el cual creamos el entorno virtual:



3. Crear entorno virtual a través de la terminal. Clic en:



4. En la terminal ingresar las siguientes líneas:

- Crear ambiente virtual

```
pip install virtualenv  
virtualenv venv
```

- Activar ambiente virtual

```
.\venv\Scripts\activate
```

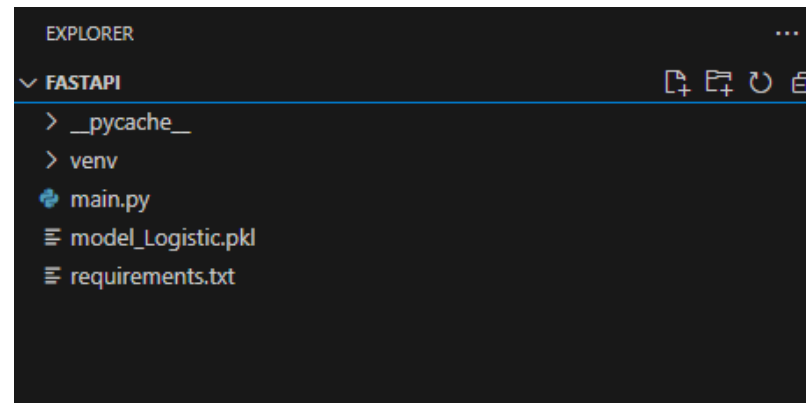
- Instalar librerías

```
pip install fastapi uvicorn pandas scikit-learn
```

- Crear documento de requerimientos

```
pip freeze > requirements.txt
```

- Crear un archivo .py al que nombraremos “main”. En el lateral izquierdo veremos los siguientes archivos:



5. En el archivo “main” ingresaremos el siguiente código para crear la instancia de FastAPI:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
import pickle
import pandas as pd

# Cargamos el modelo del archivo pickle
with open('model_Logistic.pkl', 'rb') as file:
    model, selected_features = pickle.load(file)

# Creamos una instancia de FastAPI
app = FastAPI()

# Configuración de CORS
origins = ["*"]
app.add_middleware(
    CORSMiddleware,
```

```

    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Verificación estado de servicio
@app.options("/predict")
def options_predict():
    return {"status": "ok"}

# Definimos la ruta para hacer predicciones
@app.post("/predict")
def predict(data: dict):
    print("Datos recibidos:", data) # Impresión de datos recibidos
    datos_pred = pd.DataFrame({key: [value] for key, value in data.items()})
    features = datos_pred[selected_features]
    prediccion = model.predict(features)
    print("Predicción:", prediccion.tolist()) # Impresión de resultado

    return {"prediction": prediccion.tolist()}

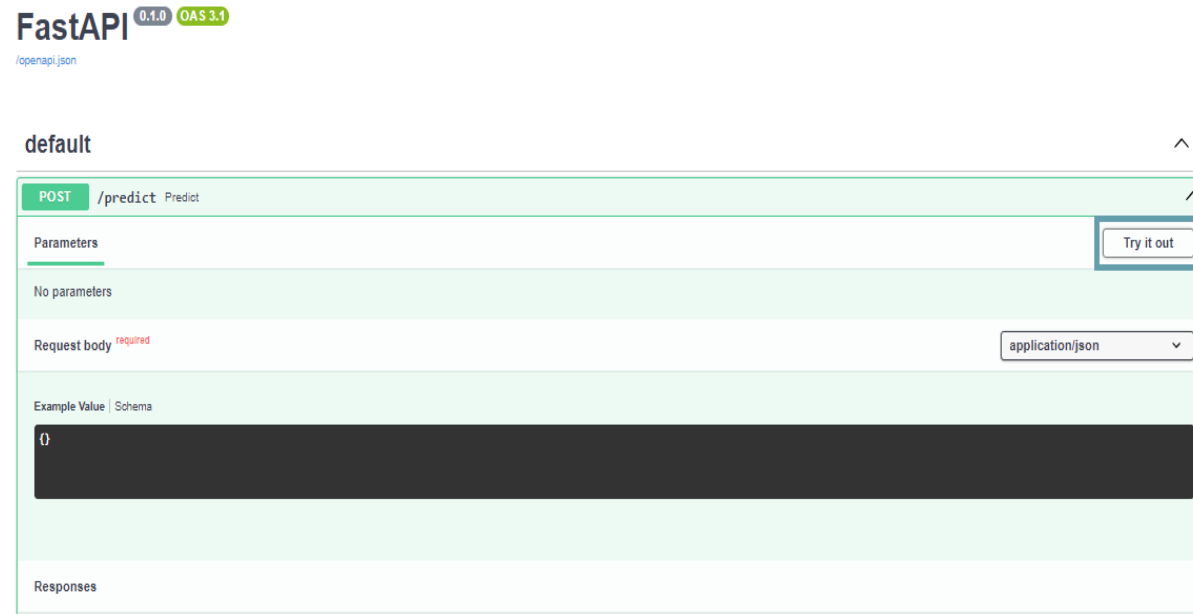
```

- En la terminal, ingresamos la siguiente línea:

```
uvicorn main:app --reload
```

De esta manera podremos probar la API desde el navegador.

6. Probaremos el servicio en local ingresando a <http://127.0.0.1:8000/docs> en el navegador. Aparecerá la interfaz interactiva Swagger UI que nos permitirá llamar a la API creada y testearla en el navegador. Clic en:



7. En el campo “Request body” ingresar los valores de las variables que el modelo seleccionó, en formato JSON y clic en “Execute”:

default

POST /predict Predict

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "BMI": 30.0,  "HighBP": "1",  "HighChol": "1",  "CholCheck": "1",  "Stroke": "0",  "HeartDiseaseorAttack": "1",  "PhysActivity": "1",  "Veggies": "1",  "HvyAlcoholConsump": "0",  "AnyHealthcare": "1",  "GenHlth": "5",  "DiffWalk": "1",  "Sex": "0",  "Age": "9",  "Education": "5",  "Income": "7"}
```

Execute

Clear

8. Verificamos en la sección “Response” que se ha realizado la predicción con los datos enviados:

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "BMI": 30.0,
    "HighBP": "1",
    "HighChol": "1",
    "CholCheck": "1",
    "Stroke": "0",
    "HeartDiseaseorAttack": "1",
    "PhysActivity": "1",
    "Veggies": "1",
    "HvyAlcoholConsump": "0",
    "AnyHealthcare": "1",
    "GenHlth": "5",
    "DiffWalk": "1",
    "Sex": "0",
    "Age": "9",
    "Education": "5",
    "Income": "7"
  }'
```

Request URL

http://127.0.0.1:8000/predict

Server response

Code Details

200

Response body

```
{
  "prediction": [
    1
  ]
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-methods: PUT,GET,HEAD,POST,DELETE,OPTIONS
access-control-allow-origin: *
content-length: 20
content-type: application/json
date: Sun, 18 Feb 2024 22:05:50 GMT
server: uvicorn
```

- Una vez realizada la verificación, procederemos a alojar nuestra API en una instancia EC2 de Amazon, esto con el fin de que el servicio pueda ser consumido , en nuestro caso a través de un formulario web.

PASO A PASO: DESPLIEGUE EN AMAZON EC2

- En ingresamos nuestro usuario y contraseña. En la página de inicio de la consola buscamos EC2 y damos clic:



2. Aparecerá la siguiente pantalla, y clic en:

Recursos

EC2 Global view

Actualmente, utiliza los siguientes recursos de Amazon EC2 en la región EE.UU. Este (Ohio):

Instancias (en ejecución)	2	Balanceadores de carga	0	Direcciones IP elásticas	0
Grupos de escalamiento automático	0	Grupos de seguridad	3	Grupos de ubicación	0
Hosts dedicados	0	Instancias	2	Instantáneas	0
Pares de claves	1	Volúmenes	2		

Lanzar la instancia

Para comenzar, lance una instancia de Amazon EC2, que es un servidor virtual en la nube.

Lanzar la instancia

Migrar un servidor

Nota: Sus instancias se lanzarán en la región EE.UU. Este (Ohio)

Instance alarms

View in CloudWatch

0 in alarm

0 OK

0 insufficient data

Instances in alarm

Estado del servicio

Panel de AWS Health

Región
EE.UU. Este (Ohio)

Zonas

Nombre de la zona	ID de la zona
us-east-2a	use2-az1
us-east-2b	use2-az2
us-east-2c	use2-az3

Habilitar zonas adicionales

3. Ingresaremos los datos señalados:

Amazon EC2 le permite crear máquinas virtuales, o instancias, que se ejecutan en la nube de AWS. Comience rápidamente siguiendo los sencillos pasos que se indican a continuación.

Nombre y etiquetas [Información](#)

Nombre [Agregar etiquetas adicionales](#)

▼ Imágenes de aplicaciones y sistemas operativos (Imagen de máquina de Amazon) [Información](#)

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

Recientes | **Inicio rápido**

macOS


Ubuntu


Windows


Red Hat


SUSE Linux


[Buscar más AMI](#)
Inclusión de AMI de AWS, Marketplace y la comunidad

Imágenes de máquina de Amazon (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type	Apto para la capa gratuita ▼
ami-05fb0b8c1424f266b (64 bits (x86)) / ami-0748d13ffbc370c2b (64 bits (Arm))	
Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs	

▼ Par de claves (inicio de sesión) Información

Puede utilizar un par de claves para conectarse de forma segura a la instancia. Asegúrese de que tiene acceso al par de claves seleccionado antes de lanzar la instancia.

Nombre del par de claves - obligatorio

eloarsa

↻ Crear un nuevo par de claves

▼ Configuraciones de red Información

Editar

Red Información

vpc-0ecb18ca4516c9171

Subred Información

Sin preferencias (subred predeterminada en cualquier zona de disponibilidad)

Asignar automáticamente la IP pública Información

Habilitar

Firewall (grupos de seguridad) Información

Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

☒ Crear grupo de seguridad

☐ Seleccionar un grupo de seguridad existente

Crearemos un nuevo grupo de seguridad denominado "launch-wizard-2" con las siguientes reglas:

☒ Permitir el tráfico de SSH desde

Ayuda a establecer conexión con la instancia

Cualquier lugar

0.0.0.0/0

☒ Permitir el tráfico de HTTPS desde Internet

Para configurar un punto de enlace, por ejemplo, al crear un servidor web

☒ Permitir el tráfico de HTTP desde Internet

Para configurar un punto de enlace, por ejemplo, al crear un servidor web

⚠ Las reglas con origen 0.0.0.0/0 permiten que todas las direcciones IP tengan acceso a la instancia. Le recomendamos que configure las reglas del grupo de seguridad para permitir el acceso únicamente desde direcciones IP conocidas.

Y creamos la instancia:

▼ Configurar almacenamiento

Información

Avanzado

1x 8 GiB gp2 Volumen raíz (Sin cifrar)

Los clientes que cumplan los requisitos de la capa gratuita pueden obtener hasta 30 GB de almacenamiento magnético o de uso general (SSD) de EBS

Agregar un nuevo volumen

La AMI seleccionada contiene más volúmenes de almacén de instancias de los que permite la instancia. Solo se podrá obtener acceso desde la instancia a los primeros 0 volúmenes de almacén de instancias de la AMI

Haga clic en actualizar para ver la información de la copia de seguridad
Las etiquetas que asigne determinan si alguna política de Data Lifecycle Manager realizará una copia de seguridad de la instancia.

0 x sistemas de archivos

Editar

▼ Resumen

Número de instancias

Información

1

Imagen de software (AMI)

Canonical, Ubuntu, 22.04 LTS, ...más información

ami-05fb0b8c1424f266b

Tipo de servidor virtual (tipo de instancia)

t2.micro

Firewall (grupo de seguridad)

Nuevo grupo de seguridad

Almacenamiento (volúmenes)

Volúmenes: 1 (8 GiB)

Nivel gratuito: El primer año incluye

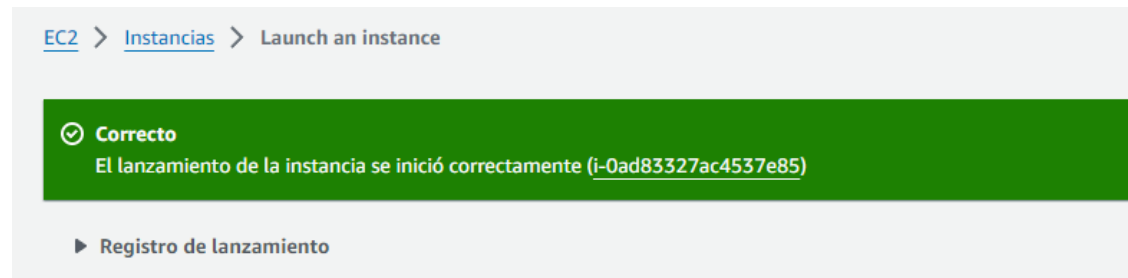
750 horas de uso de instancias t2.micro (o t3.micro en las regiones en las que t2.micro no esté disponible) en las AMI del nivel gratuito al mes, 30 GiB de almacenamiento de EBS, 2 millones de E/S, 1 GB de instantáneas y 100 GB de ancho de banda a Internet.

Cancelar

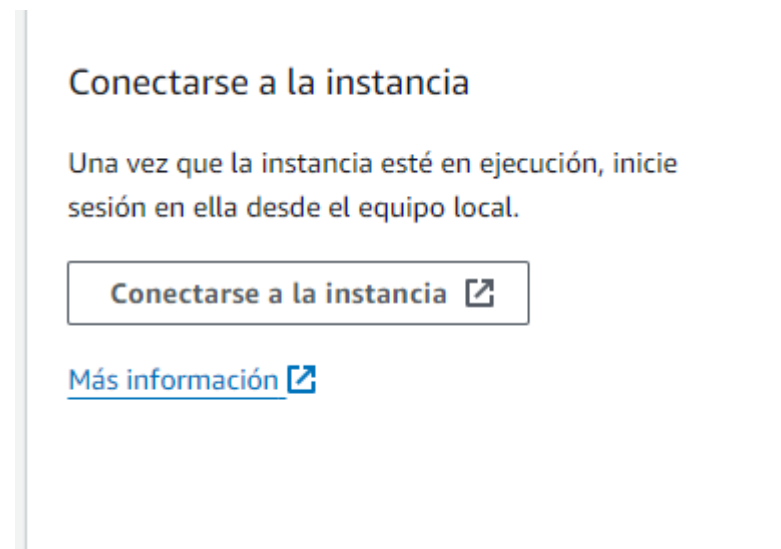
Lanzar instancia

Revisar comandos

4. Si la instancia se ha lanzado de manera exitosa, aparecerá el siguiente mensaje:



5. Nos conectamos a la instancia creada:



6. Veremos la siguiente pantalla:

[EC2](#) > [Instancias](#) > [i-0ad83327ac4537e85](#) > Conectarse a la instancia

Conectarse a la instancia [Información](#)

Conéctese a la instancia i-0ad83327ac4537e85 (fastapi) mediante cualquiera de estas opciones


Conexión de la instancia EC2

Administrador de sesiones

Cliente SSH

Consola de serie de EC2

ID de la instancia


 i-0ad83327ac4537e85 (fastapi)

Tipo de conexión

☒ Conectarse mediante la Conexión de la instancia EC2
Conéctese mediante el cliente basado en navegador de EC2 Instance Connect, con una dirección IPv4 pública.


☐ Conectarse mediante punto de conexión de EC2 Instance Connect
Conéctese mediante el cliente basado en navegador de EC2 Instance Connect, con una dirección IPv4 privada y un punto de conexión de VPC.

Dirección IP pública

 18.191.148.32

Nombre de usuario

Escriba el nombre de usuario definido en la AMI utilizada para lanzar la instancia. Si no definió un nombre de usuario personalizado, utilice el nombre de usuario predeterminado, ubuntu.

 **Nota:** En la mayoría de los casos, el nombre de usuario predeterminado, ubuntu, es correcto. Sin embargo, lea las instrucciones de uso de la AMI para comprobar si el propietario de la AMI ha cambiado el nombre de usuario predeterminado.

Cancelar

Conectar

7. Clic en la pestaña “Cliente SSH”:

EC2 > Instancias > i-0ad83327ac4537e85 > Conectarse a la instancia

Conectarse a la instancia Información

Conéctese a la instancia i-0ad83327ac4537e85 (fastapi) mediante cualquiera de estas opciones

Conexión de la instancia EC2

Administrador de sesiones

Cliente SSH

Consola de serie de EC2

ID de la instancia
i-0ad83327ac4537e85 (fastapi)

1. Abra un cliente SSH.
2. Localice el archivo de clave privada. La clave utilizada para lanzar esta instancia es eloarsa.pem
3. Ejecute este comando, si es necesario, para garantizar que la clave no se pueda ver públicamente.
chmod 400 "eloarsa.pem"
4. Conéctese a la instancia mediante su DNS público:
ec2-18-191-148-32.us-east-2.compute.amazonaws.com
Copiar la dirección de la instancia en el portapapeles

Ejemplo:

```
ssh -i "eloarsa.pem" ubuntu@ec2-18-191-148-32.us-east-2.compute.amazonaws.com
```

Nota: En la mayoría de los casos, el nombre de usuario adivinado es correcto. Sin embargo, lea las instrucciones de uso de la AMI para comprobar si el propietario de la AMI ha cambiado el nombre de usuario predeterminado de la AMI.

Y copiar la dirección de la instancia que aparece en el paso 4. Ahora la instancia se encuentra en ejecución:

☐ fastapi

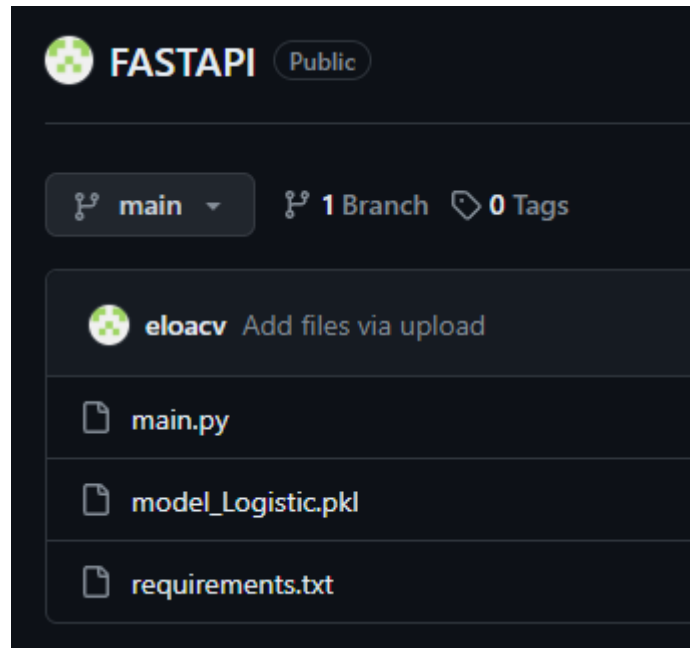
i-0ad83327ac4537e85

☒ En ejecución

t2.micro

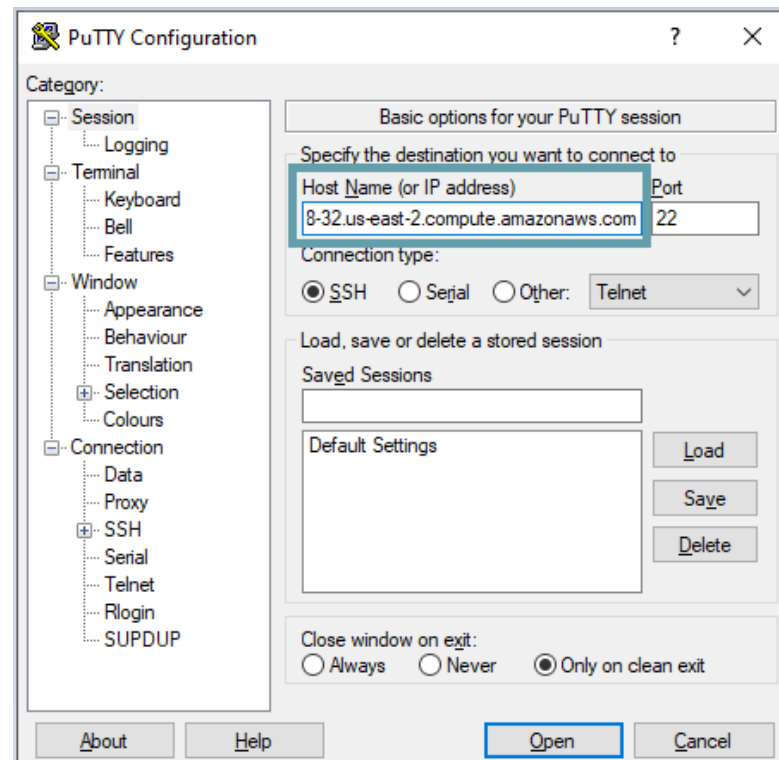
Sin embargo, en este punto nuestra instancia está vacía. Colocaremos nuestra FAsTAPI en esta instancia para que, al acceder al endpoint, podamos usar la API y realizar predicciones.

8. Creamos un repositorio en Github con los archivos :



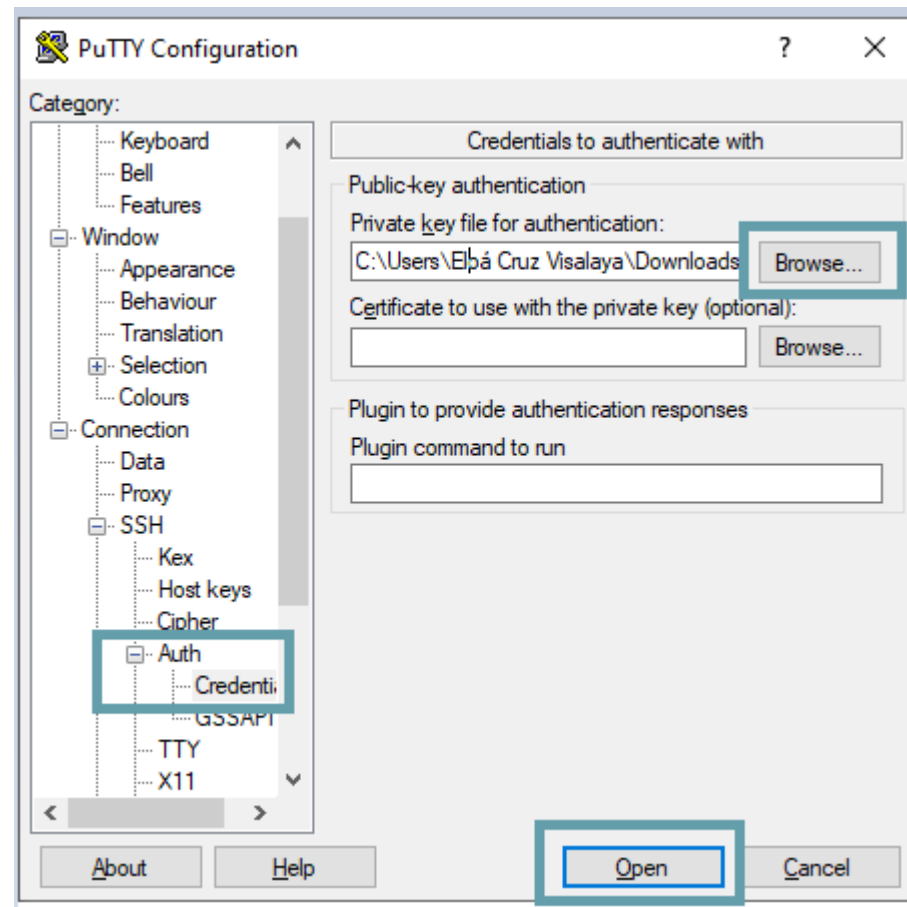
Alojados en el folder que seleccionamos para crear nuestro entorno virtual, el txt de requirements y el archivo .py “main”.

9. Para conectarnos a la instancia usaremos PuTTY¹, un cliente SSH para Windows. Abrimos la interfaz y pegamos el DNS de la instancia que copiamos en el paso 7:

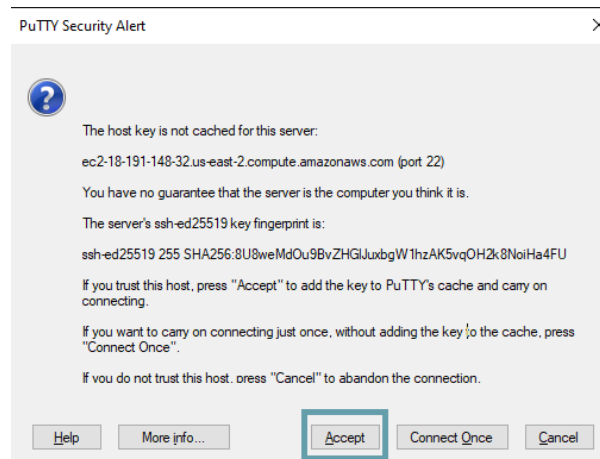


¹ <https://www.putty.org/>

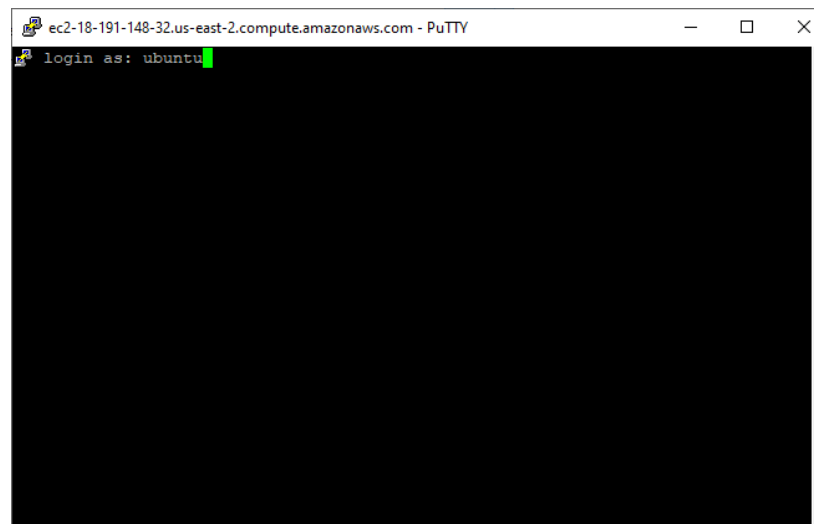
10. Nos autenticamos usando el key file que se generó en el paso 3:



11. Clic en:

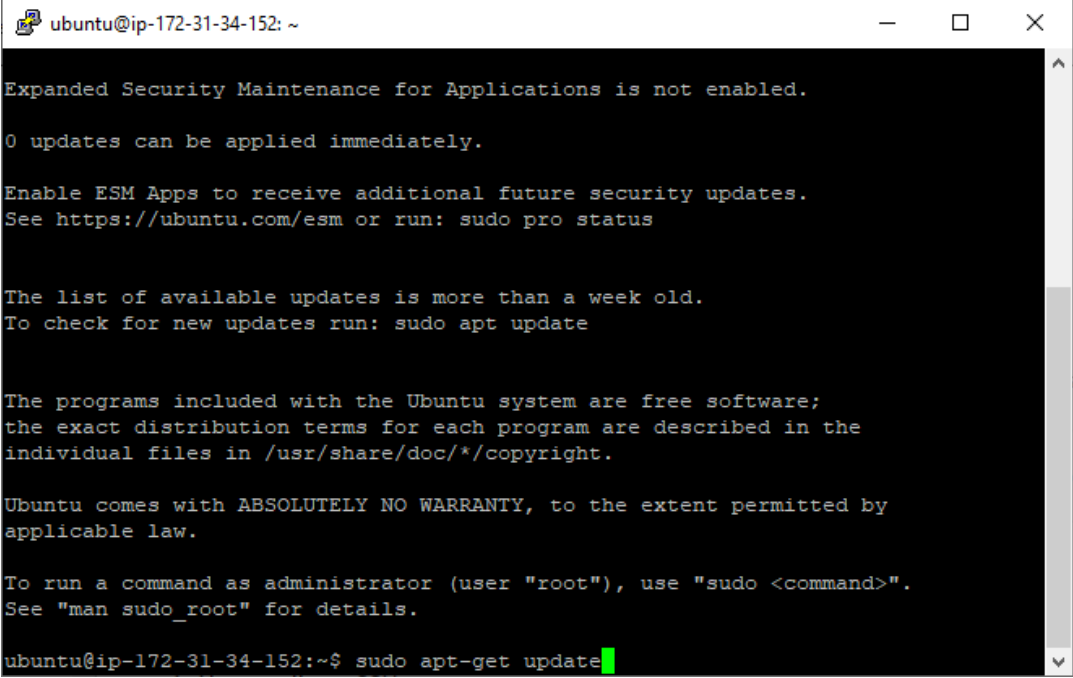


Aparece la siguiente pantalla, ingresamos “ubuntu” y enter:



12. Este terminal representa el sistema operativo Ubuntu dentro de la máquina que usa nuestra instancia. Ingresamos la siguiente línea para actualizar todos los repositorios:

sudo apt-get update



```
ubuntu@ip-172-31-34-152: ~  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-34-152:~$ sudo apt-get update
```

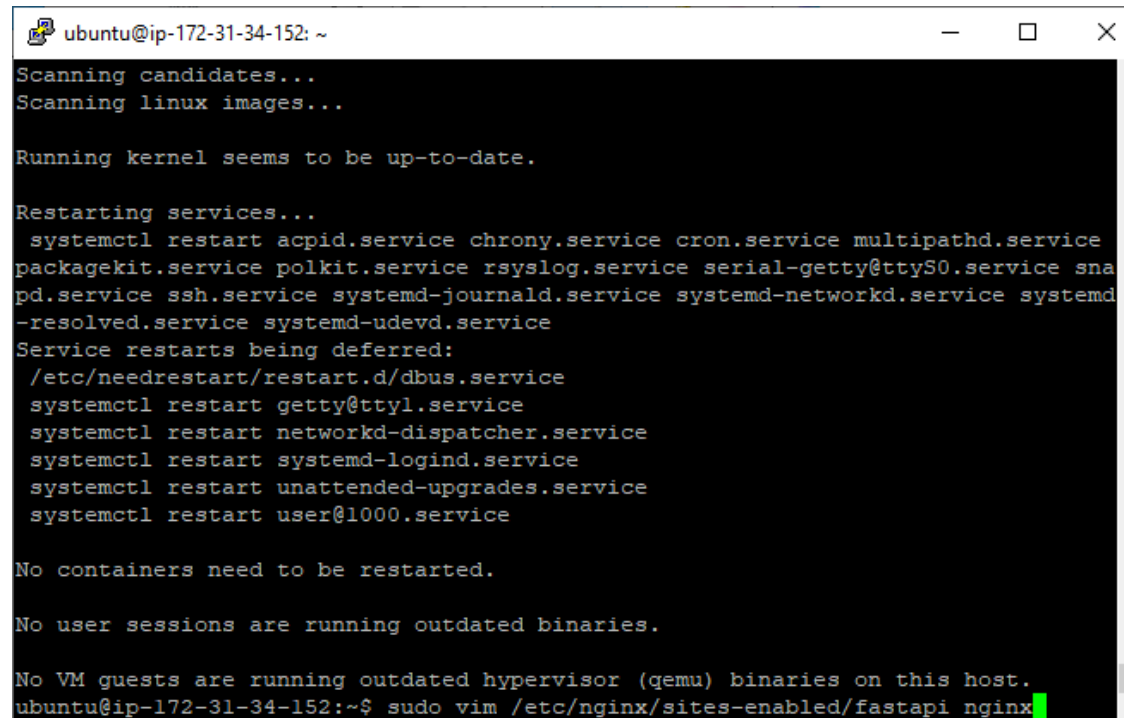
13. Instalamos python3 y nginx (que nos permite conectarnos a FastAPI) con la siguiente línea:

sudo apt install -y python3-pip nginx

```
ubuntu@ip-172-31-34-152: ~  
amd64 Packages [24.3 kB]  
Get:29 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]  
Get:30 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]  
Get:31 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]  
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [231 kB]  
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [842 kB]  
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [161 kB]  
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]  
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]  
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]  
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]  
Fetched 29.5 MB in 5s (5380 kB/s)  
Reading package lists... Done  
ubuntu@ip-172-31-34-152:~$ sudo apt install -y python3-pip nginx
```


14. Creamos un archivo de configuración en el directorio, con la siguiente línea:

sudo vim / etc/nginx/sites-enabled/fastapi_nginx

A terminal window titled 'ubuntu@ip-172-31-34-152: ~' with standard window controls. The terminal output shows the following sequence of messages:
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
systemctl restart acpid.service chrony.service cron.service multipathd.service
packagekit.service polkit.service rsyslog.service serial-getty@ttyS0.service sna
pd.service ssh.service systemd-journald.service systemd-networkd.service systemd
-resolved.service systemd-udev.service
Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
The prompt is 'ubuntu@ip-172-31-34-152:~\$' followed by the command 'sudo vim /etc/nginx/sites-enabled/fastapi_nginx' which is partially visible with a green cursor at the end.


15. Procedemos a llenar el archivo de configuración de la siguiente manera:

```
Server {  
    listen 80;  
    server_name aquí se pega la dirección IPv4 pública de nuestra instancia EC2;  
    location / {  
        proxy_pass http://127.0.0.1: 8000; (dirección del endpoint de nuestra FastAPI)  
    }  
}
```

Resumen de instancia de i-0ad83327ac4537e85 (fastapi) [Información](#)

Se ha actualizado hace less than a minute

ID de la instancia

 i-0ad83327ac4537e85 (fastapi)

Dirección IPv6

—

Dirección IPv4 pública

 18.191.148.32 [dirección abierta](#) 

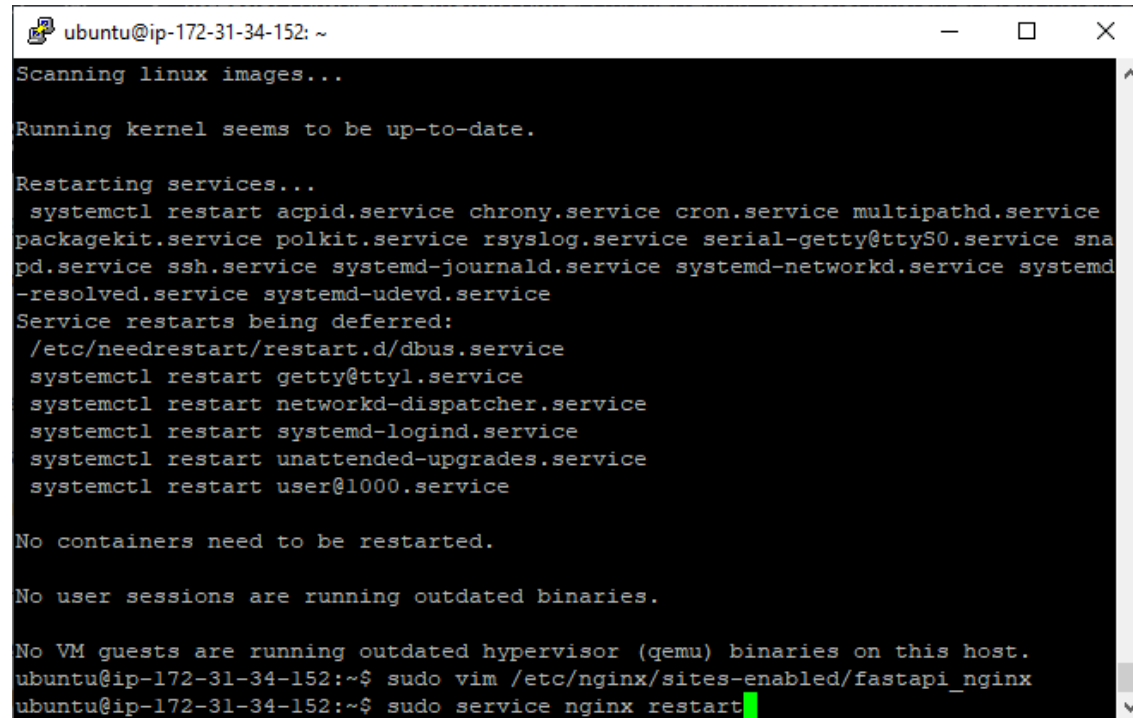
Estado de la instancia

 En ejecución

```
ubuntu@ip-172-31-34-152: ~  
server {  
    listen 80;  
    server_name 18.191.148.32;  
    location / {  
        proxy_pass http://127.0.0.1:8000;  
    }  
}
```

16. Guardamos el archivo (ESC e ingresar wq y enter) y reiniciamos el servidor nginx para que la configuración haga efecto. Para ello ingresamos esta línea:

sudo service nginx restart

A terminal window titled 'ubuntu@ip-172-31-34-152: ~' with standard window controls. The terminal output shows system checks and service restarts. It starts with 'Scanning linux images...', followed by 'Running kernel seems to be up-to-date.' and 'Restarting services...'. A long line of services is listed for restart, including acpid, chrony, cron, multipathd, packagekit, polkit, rsyslog, serial-getty@ttyS0, snapd, ssh, systemd-journald, systemd-networkd, systemd-resolved, systemd-udev, and user@1000. Below this, it says 'Service restarts being deferred:' followed by a list of deferred services: /etc/needrestart/restart.d/dbus.service, getty@tty1.service, networkd-dispatcher.service, systemd-logind.service, unattended-upgrades.service, and user@1000.service. It then states 'No containers need to be restarted.' and 'No user sessions are running outdated binaries.' and 'No VM guests are running outdated hypervisor (qemu) binaries on this host.' The final two lines show the user running 'sudo vim /etc/nginx/sites-enabled/fastapi_nginx' and then 'sudo service nginx restart', with a green cursor at the end of the second command.

```
ubuntu@ip-172-31-34-152: ~
Scanning linux images...

Running kernel seems to be up-to-date.

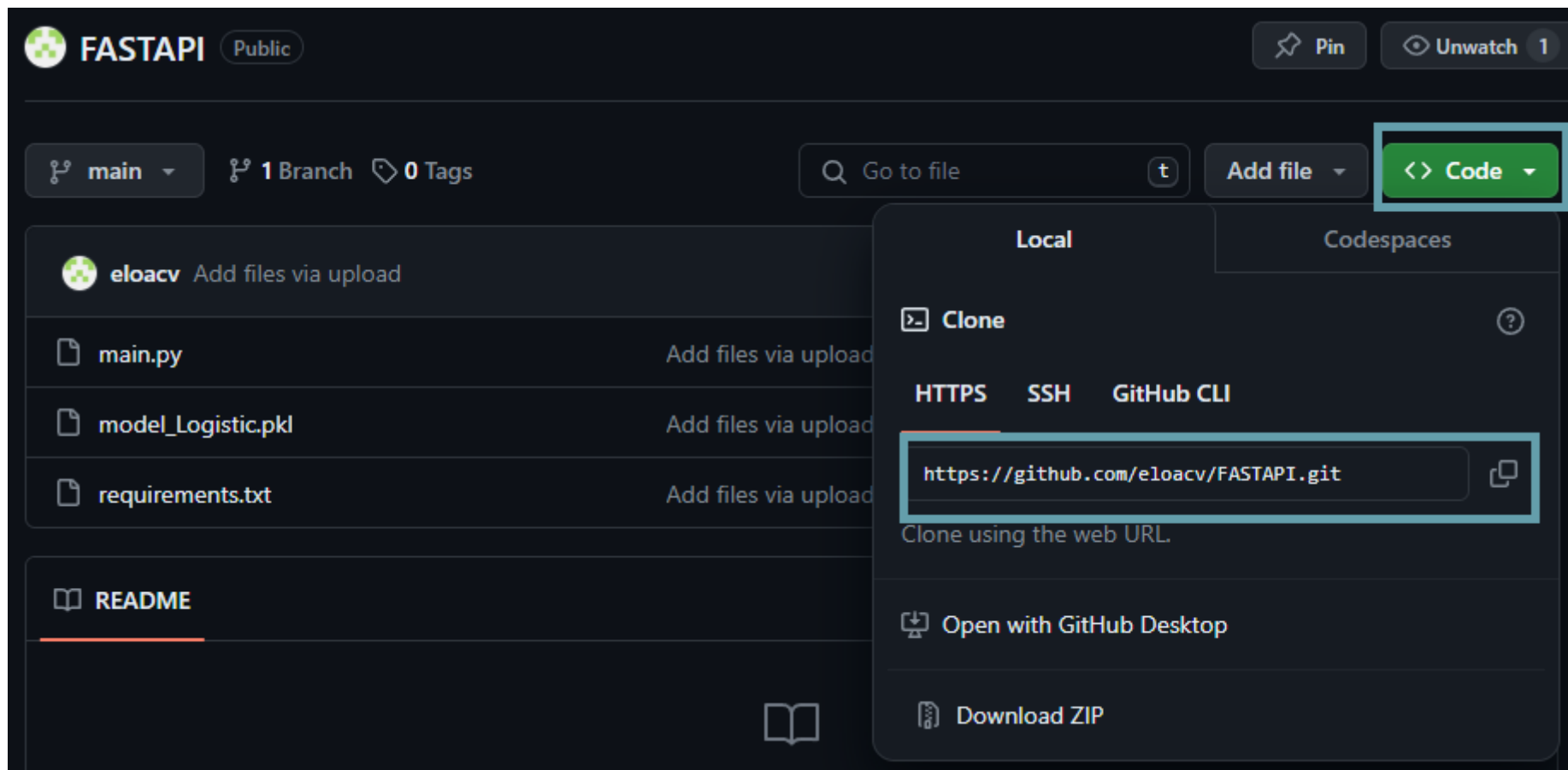
Restarting services...
systemctl restart acpid.service chrony.service cron.service multipathd.service
packagekit.service polkit.service rsyslog.service serial-getty@ttyS0.service sna
pd.service ssh.service systemd-journald.service systemd-networkd.service systemd
-resolved.service systemd-udev.service
Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-34-152:~$ sudo vim /etc/nginx/sites-enabled/fastapi_nginx
ubuntu@ip-172-31-34-152:~$ sudo service nginx restart
```

17. Ahora, clonaremos el repositorio Github creado en el paso 8. Para ello damos clic en “Code” y copiamos la url de la pestaña “HTTPS”:



18. Volvemos a PuTTY e ingresamos la siguiente línea para clonar el repositorio:

git clone url que copiamos en el paso anterior

```
ubuntu@ip-172-31-34-152: ~  
Feb 18 02:12:13 ip-172-31-34-152 nginx[353]: nginx: [emerg] unexpected end of file, expecting "}"  
Feb 18 02:12:13 ip-172-31-34-152 nginx[353]: nginx: configuration file /etc/nginx/nginx.conf test  
Feb 18 02:12:13 ip-172-31-34-152 systemd[1]: nginx.service: Control process exited, code=exited, s  
Feb 18 02:12:13 ip-172-31-34-152 systemd[1]: nginx.service: Failed with result 'exit-code'.  
Feb 18 02:12:13 ip-172-31-34-152 systemd[1]: Failed to start A high performance web server and a r  
ubuntu@ip-172-31-34-152:~$ sudo service nginx restart  
ubuntu@ip-172-31-34-152:~$ systemctl status nginx.service  
● nginx.service - A high performance web server and a reverse proxy server  
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sun 2024-02-18 02:16:44 UTC; 2s ago  
     Docs: man:nginx(8)  
  Process: 784 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited,  
 Process: 785 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0  
 Main PID: 786 (nginx)  
    Tasks: 2 (limit: 1121)  
  Memory: 2.6M  
     CPU: 25ms  
   CGroup: /system.slice/nginx.service  
           └─786 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"  
             └─787 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""  
Feb 18 02:16:44 ip-172-31-34-152 systemd[1]: Starting A high performance web server and a reverse  
Feb 18 02:16:44 ip-172-31-34-152 systemd[1]: Started A high performance web server and a reverse p  
ubuntu@ip-172-31-34-152:~$ git clone https://github.com/eloacv/FASTAPI.git
```

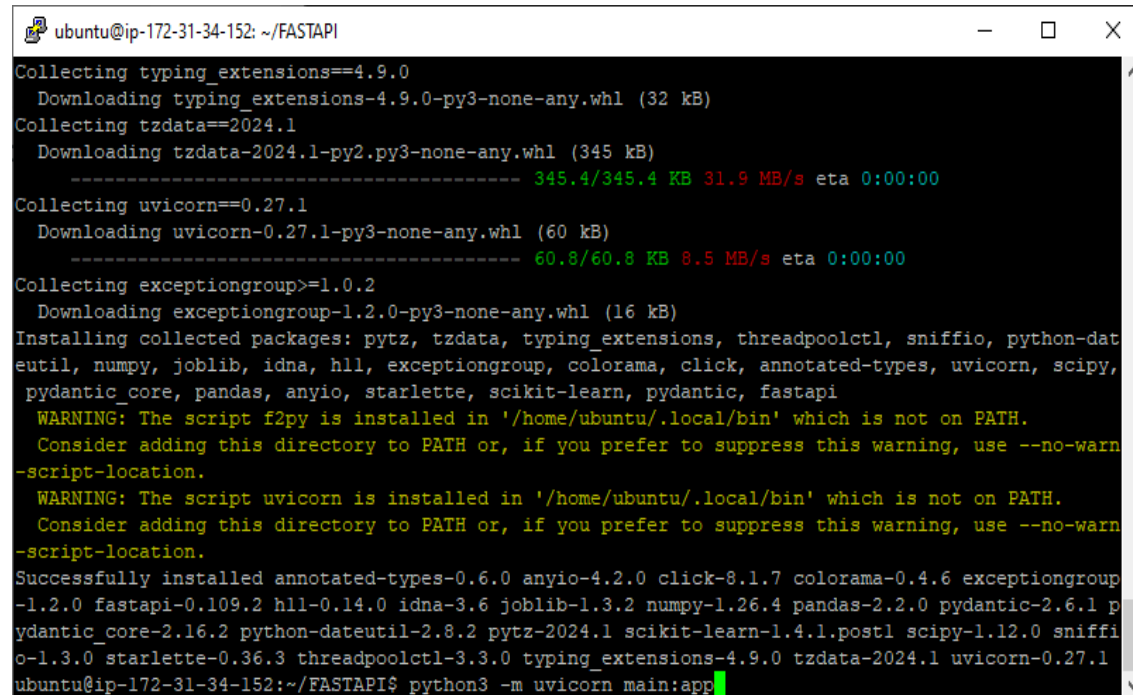
19. Se observa que se han clonado correctamente todos los archivos y procedemos a instalar los requerimientos con la siguiente línea:

pip3 install -r requirements.txt

[illegible]

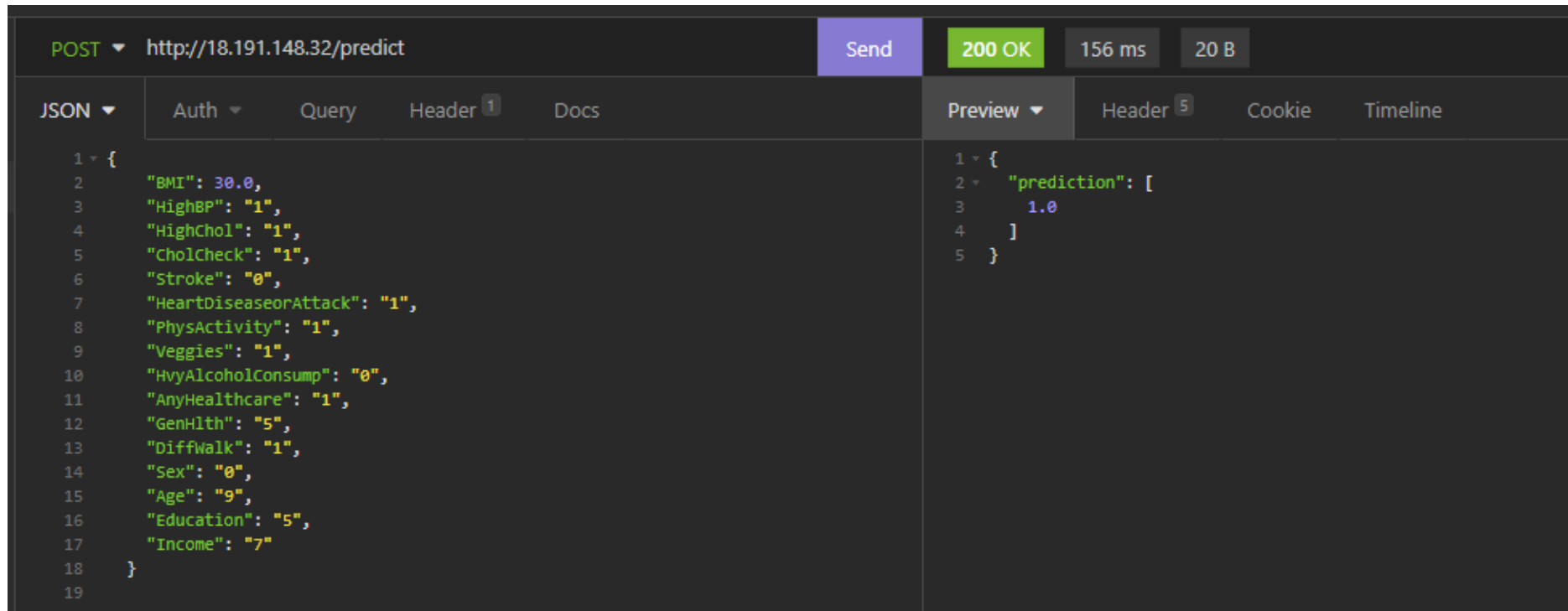
20. Una vez finalizada la instalación, procedemos a correr la app contenida en el archivo “main”, usando la siguiente línea:

python3 -m uvicorn main:app



```
ubuntu@ip-172-31-34-152: ~/FASTAPI
Collecting typing_extensions==4.9.0
  Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Collecting tzdata==2024.1
  Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
----- 345.4/345.4 KB 31.9 MB/s eta 0:00:00
Collecting uvicorn==0.27.1
  Downloading uvicorn-0.27.1-py3-none-any.whl (60 kB)
----- 60.8/60.8 KB 8.5 MB/s eta 0:00:00
Collecting exceptiongroup>=1.0.2
  Downloading exceptiongroup-1.2.0-py3-none-any.whl (16 kB)
Installing collected packages: pytz, tzdata, typing_extensions, threadpoolctl, sniffio, python-dateutil, numpy, joblib, idna, h11, exceptiongroup, colorama, click, annotated-types, uvicorn, scipy, pydantic_core, pandas, anyio, starlette, scikit-learn, pydantic, fastapi
WARNING: The script f2py is installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script uvicorn is installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed annotated-types-0.6.0 anyio-4.2.0 click-8.1.7 colorama-0.4.6 exceptiongroup-1.2.0 fastapi-0.109.2 h11-0.14.0 idna-3.6 joblib-1.3.2 numpy-1.26.4 pandas-2.2.0 pydantic-2.6.1 pydantic_core-2.16.2 python-dateutil-2.8.2 pytz-2024.1 scikit-learn-1.4.1.post1 scipy-1.12.0 sniffio-1.3.0 starlette-0.36.3 threadpoolctl-3.3.0 typing_extensions-4.9.0 tzdata-2024.1 uvicorn-0.27.1
ubuntu@ip-172-31-34-152:~/FASTAPI$ python3 -m uvicorn main:app
```

21. Usamos Insomnia² e ingresamos la IPv4 pública de la instancia seguido de la ruta para hacer predicciones y pasamos valores en formato JSON para corroborar que nuestro servicio funciona correctamente:

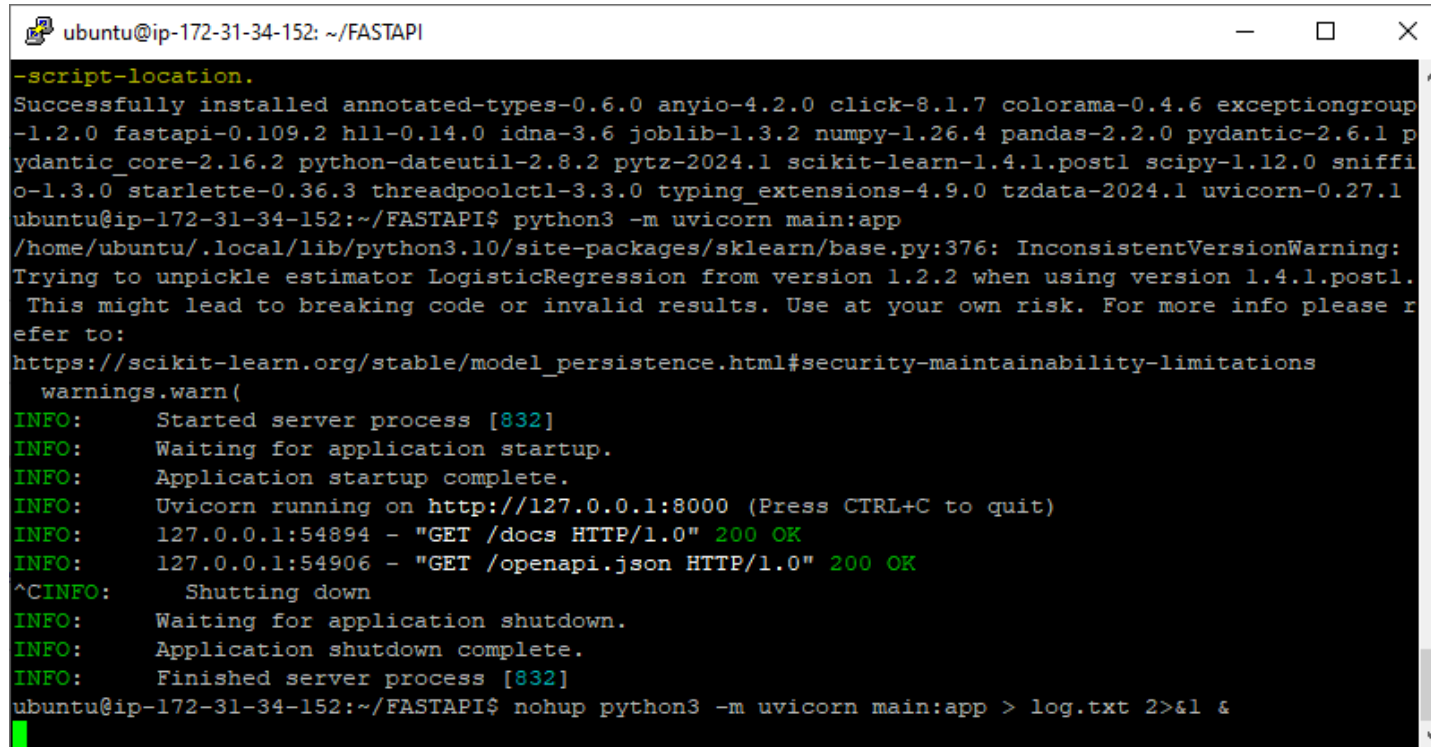


Nuestra API ahora está lista para ser usada en un formulario web.

² <https://insomnia.rest/>

22. Finalmente, quiero que mi aplicación siga ejecutándose aun cuando cierre la sesión en PuTTY por lo cual ingreso la siguiente línea:

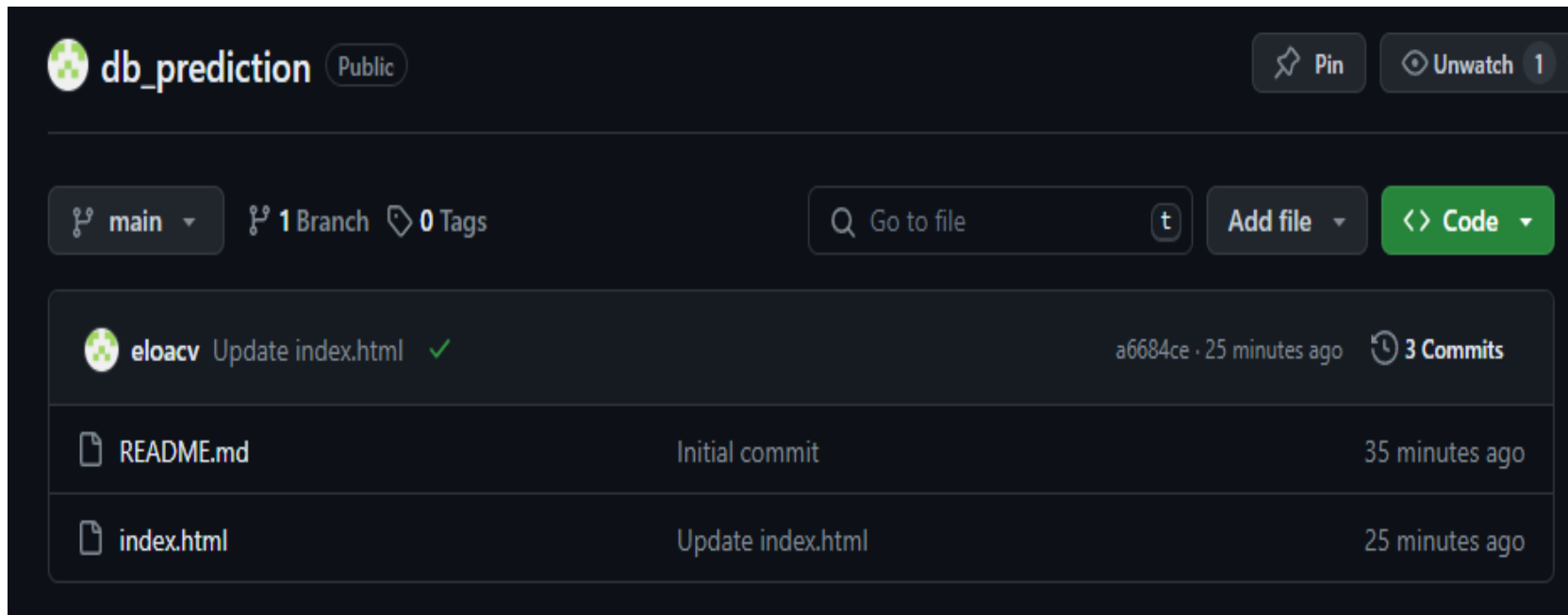
nohup python3 -m uvicorn main:app > log.txt 2>&1 &



```
ubuntu@ip-172-31-34-152: ~/FASTAPI
-script-location.
Successfully installed annotated-types-0.6.0 anyio-4.2.0 click-8.1.7 colorama-0.4.6 exceptiongroup
-1.2.0 fastapi-0.109.2 h11-0.14.0 idna-3.6 joblib-1.3.2 numpy-1.26.4 pandas-2.2.0 pydantic-2.6.1 p
ydantic_core-2.16.2 python-dateutil-2.8.2 pytz-2024.1 scikit-learn-1.4.1.post1 scipy-1.12.0 sniffi
o-1.3.0 starlette-0.36.3 threadpoolctl-3.3.0 typing_extensions-4.9.0 tzdata-2024.1 uvicorn-0.27.1
ubuntu@ip-172-31-34-152:~/FASTAPI$ python3 -m uvicorn main:app
/home/ubuntu/.local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning:
Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.4.1.post1.
This might lead to breaking code or invalid results. Use at your own risk. For more info please r
efer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
INFO:      Started server process [832]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      127.0.0.1:54894 - "GET /docs HTTP/1.0" 200 OK
INFO:      127.0.0.1:54906 - "GET /openapi.json HTTP/1.0" 200 OK
^CINFO:      Shutting down
INFO:      Waiting for application shutdown.
INFO:      Application shutdown complete.
INFO:      Finished server process [832]
ubuntu@ip-172-31-34-152:~/FASTAPI$ nohup python3 -m uvicorn main:app > log.txt 2>&1 &
```

PASO A PASO: ALOJAR FORMULARIO WEB EN GITHUB PAGES

1. Creamos un repositorio en donde alojaremos el archivo “index.html”



2. Clic en Settings, luego clic en Pages. Verificar la configuración de la fuente de publicación y clic en Save:

General

Access

- Collaborators
- Moderation options

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks
- Environments
- Codespaces
- Pages**

Security

- Code security and analysis
- Deploy keys

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Build and deployment

Source

Deploy from a branch

Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

main / (root) Save

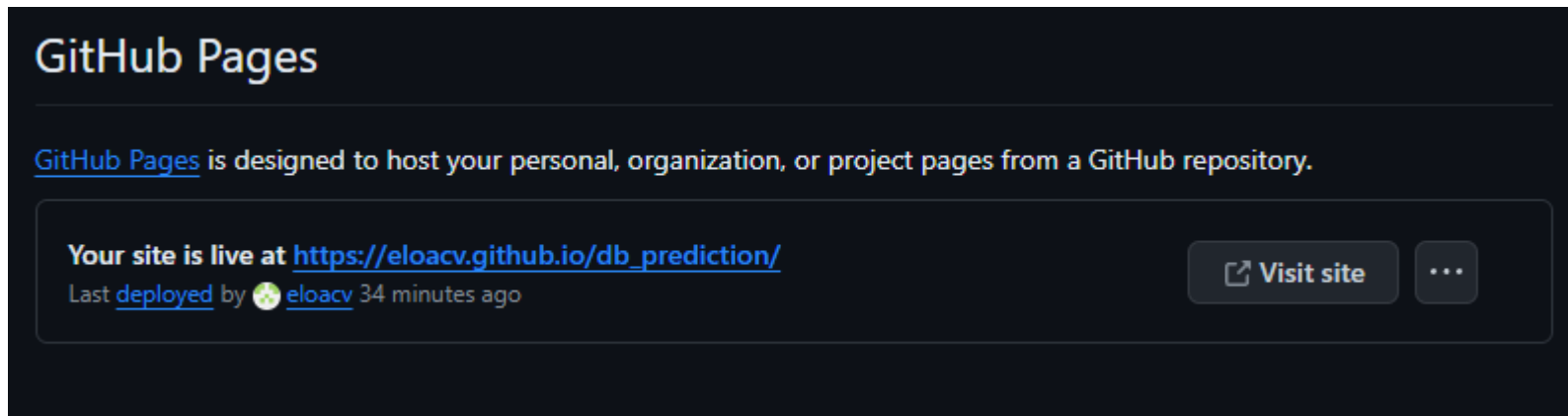
Visibility

GITHUB ENTERPRISE

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.

[Try GitHub Enterprise risk-free for 30 days](#) [Learn more about the visibility of your GitHub Pages site](#)

3. Esperamos unos minutos y aparecerá la URL del sitio:



Nota: GitHub no permite conexiones HTTP para la API. Se ha utilizado un certificado auto firmado³ (self-signed SSL) para evitar errores de mezcla de contenido (mixed content⁴). Sin embargo, es importante tener en cuenta que el uso de certificados auto firmados no es una práctica segura en entornos de producción ya que se debe contar con un certificado emitido por una autoridad de certificación reconocida para garantizar la seguridad y la autenticidad de las conexiones HTTPS.

Antes de acceder al formulario web https://eloacv.github.io/db_prediction/ , es necesario confirmar la confianza en este servidor: <https://18.191.148.32>

³ Siguiendo las instrucciones de <https://lcalcagni.medium.com/deploy-your-fastapi-to-aws-ec2-using-nginx-aa8aa0d85ec7>

⁴ <https://docs.github.com/en/pages/getting-started-with-github-pages/securing-your-github-pages-site-with-https>