```
from google.colab import files
uploaded = files.upload()
```

Choose Files dataset_med.csv

- **dataset_med.csv**(text/csv) - 93378397 bytes, last modified: 5/30/2024 - 100% done

```
Saving dataset_med.csv to dataset_med.csv
```

```
import pandas as pd

df = pd.read_csv("dataset_med.csv")
df.head()
```

er_stage	family_history	smoking_status	bmi	cholesterol_level	hypertension	asth
Stage I	Yes	Passive Smoker	29.4	199	0	
Stage III	Yes	Passive Smoker	41.2	280	1	
Stage III	Yes	Former Smoker	44.0	268	1	
Stage I	No	Passive Smoker	43.0	241	1	
Stage I	No	Passive Smoker	19.7	178	0	

```
# Drop 'id' and check structure
df = df.drop(columns=['id'])

# Basic info
print("Shape:", df.shape)
print("\nColumn types:\n", df.dtypes)
print("\nMissing values:\n", df.isnull().sum())
print("\nUnique values in target:\n", df['survived'].value_counts())
```

```
Shape: (890000, 16)

Column types:
 age                   float64
gender                 object
country                object
diagnosis_date         object
cancer_stage           object
family_history         object
smoking_status         object
bmi                   float64
cholesterol_level       int64
hypertension            int64
asthma                  int64
cirrhosis               int64
other_cancer            int64
treatment_type         object
end_treatment_date     object
survived                int64
dtype: object

Missing values:
 age                   0
gender                0
country               0
diagnosis_date        0
cancer_stage          0
family_history        0
smoking_status        0
bmi                   0
cholesterol_level     0
hypertension          0
asthma                0
cirrhosis             0
other_cancer          0
treatment_type        0
end_treatment_date    0
survived              0
dtype: int64
```

```
Unique values in target:
 survived
0    693996
1    196004
Name: count, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder

# Drop columns not useful for prediction
df = df.drop(columns=['diagnosis_date', 'end_treatment_date', 'country'])

# List of categorical columns
cat_cols = ['gender', 'cancer_stage', 'family_history', 'smoking_status', 'treatment_type']

# Label encode
le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

# Check updated dataset
df.head()
```

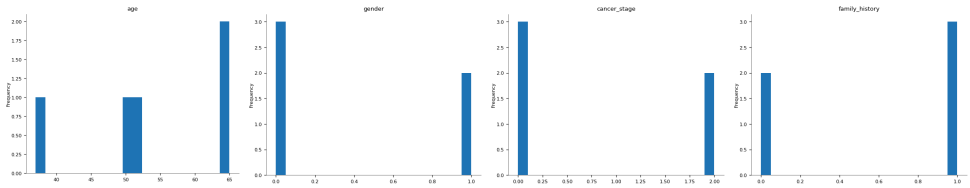
1 to 5 of 5 entries Filter

index	age	gender	cancer_stage	family_history	smoking_status	bmi	cholesterol_level	hypertension	asthma	cirrhosis	other_cancer	treatment_type	sur
0	64.0	1	0	1	3	29.4	199	0	0	1	0	0	
1	50.0	0	2	1	3	41.2	280	1	1	0	0	3	
2	65.0	0	2	1	1	44.0	268	1	1	0	0	1	
3	51.0	0	0	0	3	43.0	241	1	1	0	0	0	
4	37.0	1	0	0	3	19.7	178	0	0	0	0	1	

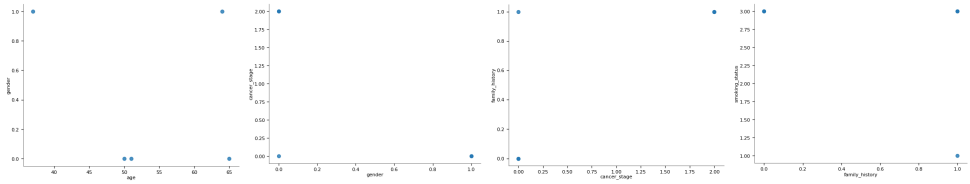
Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

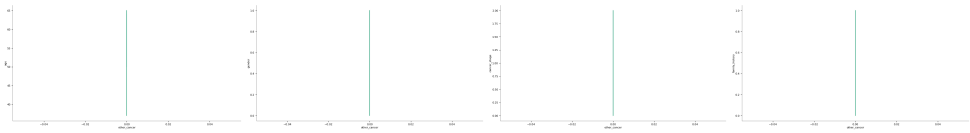
Distributions



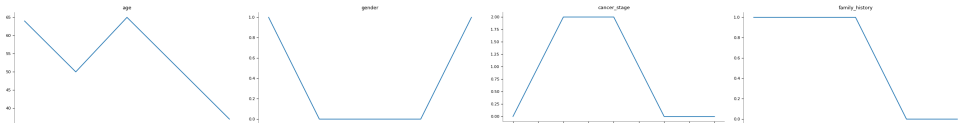
2-d distributions



Time series



Values




```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Split into features and target
X = df.drop(columns=['survived'])
y = df['survived']

# Train-test split
```

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

# Train Random Forest with class balancing
model = RandomForestClassifier(class_weight='balanced', random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluation
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
print("📉 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
✅ Accuracy: 0.7722078651685393

📊 Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.99      0.87    138639
           1       0.21      0.01      0.02     39361

    accuracy                           0.77    178000
   macro avg       0.50      0.50      0.45    178000
weighted avg       0.65      0.77      0.68    178000

📉 Confusion Matrix:
 [[137017   1622]
 [ 38925    436]]
```