```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report


df_train=pd.read_csv('CarInsurance_train.csv')


df_train.head(5)
```

	Id	Age	Job	Marital	Education	Default	Balance	HHInsurance	CarLoan	Communication	Last
0	1	32	management	single	tertiary	0	1218	1	0	telephone	
1	2	32	blue-collar	married	primary	0	1156	1	0	NaN	
2	3	29	management	single	tertiary	0	637	1	0	cellular	
3	4	25	student	single	primary	0	373	1	0	cellular	
4	5	30	management	married	tertiary	0	2694	0	0	cellular	

Next steps: Generate code with `df_train` View recommended plots New interactive sheet

```
df_train.isnull().sum()
```

	0
Id	0
Age	0
Job	19
Marital	0
Education	169
Default	0
Balance	0
HHInsurance	0
CarLoan	0
Communication	902
LastContactDay	0
LastContactMonth	0
NoOfContacts	0
DaysPassed	0
PrevAttempts	0
Outcome	3042
CallStart	0
CallEnd	0
CarInsurance	0

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Id                4000 non-null   int64 
 1   Age               4000 non-null   int64 
 2   Job               3981 non-null   object
 3   Marital           4000 non-null   object
 4   Education         3831 non-null   object
 5   Default           4000 non-null   int64 
 6   Balance           4000 non-null   int64 
 7   HHInsurance       4000 non-null   int64 
 8   CarLoan           4000 non-null   int64 
 9   Communication     3098 non-null   object
 10  LastContactDay    4000 non-null   int64 
 11  LastContactMonth  4000 non-null   object
 12  NoOfContacts      4000 non-null   int64 
 13  DaysPassed        4000 non-null   int64 
 14  PrevAttempts      4000 non-null   int64 
 15  Outcome           958 non-null    object
 16  CallStart         4000 non-null   object
 17  CallEnd           4000 non-null   object
 18  CarInsurance      4000 non-null   int64 
dtypes: int64(11), object(8)
memory usage: 593.9+ KB
```

```
categorical_columns = ['Job', 'Education', 'Communication', 'Outcome']
df_train[categorical_columns] = df_train[categorical_columns].fillna('Unknown')
df_train.fillna(df_train.median(numeric_only=True), inplace=True)
```

```
print("Data types before encoding:\n", df_train.dtypes)

# Convert categorical columns to dummy variables
df_train = pd.get_dummies(df_train, columns=['Job', 'Marital', 'Education', 'Communication', 'Outcome', 'Las

# Drop irrelevant columns
df_train = df_train.drop(columns=['Id', 'CallStart', 'CallEnd'])

# Check again to ensure all features are numeric
print("Data types after encoding:\n", df_train.dtypes)
```

```
Balance              int64
HHInsurance          int64
CarLoan              int64
Communication       object
LastContactDay       int64
LastContactMonth    object
NoOfContacts         int64
DaysPassed           int64
PrevAttempts         int64
Outcome             object
CallStart           object
CallEnd             object
CarInsurance         int64
dtype: object
Data types after encoding:
```

```
DaysPassed                int64
PrevAttempts              int64
CarInsurance              int64
Job_admin.                 bool
Job_blue-collar            bool
Job_entrepreneur           bool
Job_housemaid              bool
Job_management             bool
Job_retired                bool
Job_self-employed          bool
Job_services               bool
Job_student                bool
Job_technician             bool
Job_unemployed             bool
Marital_married            bool
Marital_single             bool
Education_primary          bool
Education_secondary        bool
Education_tertiary         bool
Communication_cellular     bool
Communication_telephone    bool
Outcome_failure            bool
Outcome_other              bool
Outcome_success            bool
LastContactMonth_aug       bool
LastContactMonth_dec       bool
LastContactMonth_feb       bool
LastContactMonth_jan       bool
LastContactMonth_jul       bool
LastContactMonth_jun       bool
LastContactMonth_mar       bool
LastContactMonth_may       bool
LastContactMonth_nov       bool
LastContactMonth_oct       bool
LastContactMonth_sep       bool
dtype: object
```

```
X = df_train.drop(columns=['CarInsurance']).astype(int)
y = df_train['CarInsurance']


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

# Train logistic regression model
log_reg = LogisticRegression(max_iter=1000)  # Increased max_iter to ensure convergence
log_reg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

LogisticRegression(max_iter=1000)

```
y_pred = log_reg.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.73

Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.87      0.80       484
           1       0.72      0.50      0.59       316

    accuracy                           0.73       800
   macro avg       0.73      0.69      0.69       800
weighted avg       0.73      0.73      0.72       800
```

```
df_test = pd.read_csv('CarInsurance_test.csv')


df_test[categorical_columns] = df_test[categorical_columns].fillna('Unknown')
df_test.fillna(df_test.median(numeric_only=True), inplace=True)


df_test = pd.get_dummies(df_test, columns=['Job', 'Marital', 'Education', 'Communication', 'Outcome', 'Last

# Align columns of test data with training data
df_test = df_test.reindex(columns=X.columns, fill_value=0)


predictions = log_reg.predict(df_test)

# Add predictions to test dataset
df_test['CarInsurance'] = predictions

# Save predictions to a new file
df_test.to_csv('CarInsurance_predictions.csv', index=False)
print("Predictions saved to 'CarInsurance_predictions.csv'")
```

```
Predictions saved to 'CarInsurance_predictions.csv'
```

```
df=pd.read_csv('CarInsurance_predictions.csv')


df.head()
```

Start coding or generate with AI.