Classifier model

car_sales_missing=pd.read_csv("car-sales-extended-missing-data.csv")
car_sales_missing

 # Check the score of the Ridge model on test data
model.score(x_test,y_test)

```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

np.random.seed(42)

x=heart_disease.drop("target",axis=1)
y=heart_disease["target"]

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

clf=RandomForestClassifier()
clf.fit(x_train,y_train)

#Make some predictions
y_preds=clf.predict(x_test)

#Evaluate the classifier
print("Classifier metrics on the test set")
print(f"accuracy:{accuracy_score(y_test,y_preds)*100:.2f}%")
print(f"precision:{precision_score(y_test,y_preds)}")
print(f"recall:{recall_score(y_test,y_preds)}")
print(f"f1:{f1_score(y_test,y_preds)}")
```

Regression model

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

np.random.seed(42)

new_x=boston_df.drop("target",axis=1)
new_y=boston_df["target"]

x_train,x_test,y_train,y_test=train_test_split(new_x,new_y,test_size=0.2)

model=RandomForestRegressor()
```

```
model.fit(x_train,y_train)
# Make prediction using our regression model
y_preds=model.predict(x_test)
#evaluate the regression model
print("regression model metrics on the test set")
print(f"R^2:{r2_score(y_test,y_preds)}")
print(f"MAE:{mean_absolute_error(y_test,y_preds)}")
print(f"MSE:{mean_squared_error(y_test,y_preds)}")
```

```
from sklearn.linear_model import LinearRegression

from sklearn.linear_model import LogisticRegression
```

```
from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
```
11:05 PM

```
from sklearn import svm
```
11:06 PM

For svm

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)

# Fitting the classifier into the Training set

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_Train, Y_Train)

# Predicting the test set results

Y_Pred = classifier.predict(X_Test)
```