Predicting Insurance Charges Using Machine Learning

Stage 1: Identifying the Problem Type

The input data is numeric and structured (Excel format), which makes it a suitable candidate for a machine learning approach.

Stage 2: Supervised Learning

We have clearly defined inputs and a known output. This makes it a case of supervised learning.

Stage 3: Type of Prediction

The goal is to predict a continuous value — the insurance charges — so we should apply a regression algorithm.

Dataset Overview

Column	Description
age	Age of the individual
sex	Gender (nominal)
bmi	Body Mass Index
children	Number of children
smoker	Smoking status (nominal)
charges	Medical insurance charges (target variable)

Input and Output Parameters

Input Features: age, sex, bmi, children, smoker

Nominal Features: sex, smoker

Target (Dependent) Variable: charges

Preprocessing

Since sex and smoker are nominal categorical variables, we need to convert them into numeric format using one-hot encoding (pd.get_dummies in Python).

After encoding, the dataset includes:

age
bmi
children
charges
sex_female, sex_male
smoker_no, smoker_yes

Model Creation and Evaluation

1. Multiple Linear Regression

R^2 Score: 0.7891

2. Support Vector Machine (SVM)

Kernel: RBF
Gamma: scale
C: 3000
R^2 Score: 0.8646

SVM RESULTS:

kernel	gamma	C	r2_score
rbf	scale	3000	0.864625669
rbf	auto	3000	0.864625669
poly	scale	2000	0.858343302
poly	auto	2000	0.858343302
poly	scale	3000	0.858085327
poly	auto	3000	0.858085327
poly	scale	1000	0.854651559
poly	auto	1000	0.854651559
rbf	scale	2000	0.85407307
rbf	auto	2000	0.85407307
rbf	scale	1000	0.810719571
rbf	auto	1000	0.810719571
linear	scale	1000	0.764839482
linear	auto	1000	0.764839482
linear	scale	2000	0.743935278

linear	auto	2000	0.743935278
linear	scale	3000	0.741337012
linear	auto	3000	0.741337012
linear	scale	100	0.628963203
linear	auto	100	0.628963203
poly	scale	100	0.616469835
poly	auto	100	0.616469835
sigmoid	scale	100	0.52684154
sigmoid	auto	100	0.52684154
linear	scale	10	0.462426338
linear	auto	10	0.462426338
rbf	scale	100	0.319664545
rbf	auto	100	0.319664545
sigmoid	scale	1000	0.212045419
sigmoid	auto	1000	0.212045419
sigmoid	scale	10	0.039440121
sigmoid	auto	10	0.039440121
poly	scale	10	0.038625187
poly	auto	10	0.038625187
rbf	scale	10	-0.0323806
rbf	auto	10	-0.0323806
sigmoid	scale	2000	-0.621621307
sigmoid	auto	2000	-0.621621307
sigmoid	scale	3000	-2.143154201
sigmoid	auto	3000	-2.143154201

3. Decision Tree

Kernel: squared_error

Gamma: sqrt

C: best

R^2 Score: 0.7839

Decision Tree Result:

criterion	max_features	splitter	r2_score
squared_error	sqrt	best	0.783928
squared_error	log2	best	0.783928
friedman_mse	sqrt	best	0.783928
friedman_mse	log2	best	0.783928
poisson	sqrt	best	0.73957
poisson	log2	best	0.73957
absolute_error		best	0.731008

squared_error		random	0.726295
friedman_mse		random	0.726295
poisson		random	0.716054
absolute_error		random	0.713449
absolute_error	sqrt	best	0.707214
absolute_error	log2	best	0.707214
absolute_error	sqrt	random	0.706453
absolute_error	log2	random	0.706453
poisson		best	0.694577
squared_error		best	0.690923
friedman_mse		best	0.690923
poisson	sqrt	random	0.683399
poisson	log2	random	0.683399
squared_error	sqrt	random	0.679295
squared_error	log2	random	0.679295
friedman_mse	sqrt	random	0.679295
friedman_mse	log2	random	0.679295

4. Random Forest

Criterion: absolute_error

Max Depth/Leaf Criteria: sqrt

Number of Estimators: 200

R^2 Score: 0.8699

Random Forest Result:

criterion	max_features	n_estimators	r2_score
absolute_error	**sqrt**	**200**	**0.869889**
absolute_error	log2	200	0.869889
absolute_error	sqrt	150	0.868543
absolute_error	log2	150	0.868543
squared_error	sqrt	200	0.866721
squared_error	log2	200	0.866721
friedman_mse	sqrt	200	0.866721
friedman_mse	log2	200	0.866721
absolute_error	sqrt	100	0.866662
absolute_error	log2	100	0.866662
squared_error	sqrt	150	0.866346
squared_error	log2	150	0.866346
friedman_mse	sqrt	150	0.866346
friedman_mse	log2	150	0.866346
squared_error	sqrt	100	0.866172

squared_error	log2	100	0.866172
friedman_mse	sqrt	100	0.866172
friedman_mse	log2	100	0.866172
absolute_error	sqrt	50	0.862123
absolute_error	log2	50	0.862123
poisson	sqrt	200	0.861076
poisson	log2	200	0.861076
squared_error	sqrt	50	0.860384
squared_error	log2	50	0.860384
friedman_mse	sqrt	50	0.860384
friedman_mse	log2	50	0.860384
poisson	sqrt	150	0.859943
poisson	log2	150	0.859943
poisson	sqrt	100	0.859143
poisson	log2	100	0.859143
absolute_error		50	0.857652
absolute_error		150	0.856914
absolute_error		100	0.85657
absolute_error		200	0.85648
poisson	sqrt	50	0.854826
poisson	log2	50	0.854826
squared_error		150	0.852924
friedman_mse		150	0.852924
squared_error		200	0.852676
friedman_mse		200	0.852676
squared_error		100	0.852465
friedman_mse		100	0.852465
poisson		200	0.852448
squared_error		50	0.851916
friedman_mse		50	0.851916
poisson		150	0.851427
poisson		100	0.850502
poisson		50	0.849567

Conclusion

Among the models tested, Random Forest produced the best result with an R^2 score of 0.8699, indicating it is the most effective model for predicting insurance charges with this dataset. Model is created and deployed.