Problem Statement: Predict health insurance premiums (medical charges) using demographic, lifestyle, and medical history variables from the **Insurance_Prediction** table (1M records).

Business Constraint:

- First **700k** → model training, Next **200k** → validation, Last **100k** → production/live scoring
- Same preprocessing must run during **training and prediction** and Deliverable must support **batch prediction or real-time API**

Target: charges (continuous → regression problem)

Available features:

- Age, Gender, BMI, Children, Smoker, Region, Occupation, Medical history, Family medical history, Exercise frequency, Coverage level

[2]:

	age	gender	bmi	children	smoker	region	medical_history	family_medical_history	exercise_frequency	occupation	coverage_level	charges
0	46.0	male	21.45	5.0	yes	southeast	Diabetes	None	Never	Blue collar	Premium	20460.307669
1	25.0	female	25.38	2.0	yes	northwest	Diabetes	High blood pressure	Occasionally	White collar	Premium	20390.899218
2	38.0	male	44.88	2.0	yes	southwest	None	High blood pressure	Occasionally	Blue collar	Premium	20204.476302
3	25.0	male	19.89	0.0	no	northwest	None	Diabetes	Rarely	White collar	Standard	11789.029843
4	49.0	male	38.21	3.0	yes	northwest	Diabetes	High blood pressure	Rarely	White collar	Standard	19268.309838

---

3. Data Split Strategy (Business Requirement)

Dataset	Records	Purpose
Train	7,00,000	Model learning
Validation	2,00,000	Model evaluation
Production	1,00,000	Real-world prediction

[5]:
```
train = df.iloc[:700000].reset_index(drop=True)
val   = df.iloc[700000:900000].reset_index(drop=True)
prod  = df.iloc[900000:].reset_index(drop=True)

print(train.shape, val.shape, prod.shape)
```
```
(700000, 12) (200000, 12) (100000, 12)
```

---

4. Data Cleaning & Preprocessing

Performed inside **preprocessing.py** to guarantee consistency.

Missing Value Treatment

Numerical

- age, children → filled with median
- children rounded and cast to integer

[13]:
```
#filling the value as median for age and children
for i in ['age','children']:
    median_val=train[i].median()
    train[i]=train[i].fillna(median_val)

#rounding the values in column children
train['children']=train['children'].round()

#changing the data type to astype(Int64)
train['children']=train['children'].astype('Int64')
```

Categorical

- gender → mode
- medical_history, family_medical_history → "No_Record"
- exercise_frequency, occupation → "Unknown"

```
[17]: # data entry issue and user skipped, Filling with mode = “male” does not change medical meaning much.
train['gender']=train['gender'].fillna(train['gender'].mode()[0])

[18]: # missing value might mean: no tests done, No diagnosis, not recorded, hence filling with No_Record
train['medical_history']=train['medical_history'].fillna('No_Record')

[19]: train['family_medical_history']=train['family_medical_history'].fillna('No_Record')

[20]: # Missing value mean : Person didn’t answer, Person unsure, Not recorded So missing is not equal to any existing category, hence unknown
train['exercise_frequency']=train['exercise_frequency'].fillna('Unknown')

[21]: train['occupation']=train['occupation'].fillna('Unknown')
```

Encoding Strategy

Feature	Encoding Type
Gender, Smoker	Binary
exercise_frequency, coverage_level	Ordinal
medical_history, family_medical_history, Occupation	One-Hot

```
[26]: train['gender']=train['gender'].map({'male':0, 'female':1})

[27]: train['smoker']=train['smoker'].map({'yes':1,'no':0})

[28]: train=pd.get_dummies(train, columns=['region', 'medical_history','family_medical_history', 'occupation'], prefix=['region','med_h','fam_med_h','occ'], dr

[29]: train['exercise_frequency']=train['exercise_frequency'].map({'never':0, 'rarely':1, 'occasionally':2, 'frequently':3})

[30]: train['coverage_level']=train['coverage_level'].map({'basic':0, 'standard':1, 'premium':2})
```

Scaling

StandardScaler applied to:

- age, bmi, children

Purpose: Stable coefficients, Compatible with linear and regularized models and Consistent preprocessing pipeline

---

5. Feature Selection Approach

1. Target correlation analysis

Correlation with target

```
[43]: corr=X_train.join(y_train).corr()['charges'].sort_values(ascending=False)
      corr
```

```
[43]: charges                          1.000000
      smoker                           0.565662
      coverage_level                   0.463364
      med_h_heart disease              0.393801
      fam_med_h_heart disease          0.392399
      exercise_frequency               0.164179
      occ_white collar                 0.128043
      bmi                              0.103391
      children                         0.076237
      age                              0.060833
      region_southeast                 0.001091
      occ_unknown                      0.000646
      region_northwest                -0.026939
      region_southwest                -0.040140
      occ_student                     -0.062523
      gender                          -0.112955
      occ_unemployed                  -0.127895
      fam_med_h_high blood pressure   -0.131265
      med_h_high blood pressure       -0.133094
      med_h_no_record                 -0.261450
      fam_med_h_no_record             -0.262272
      Name: charges, dtype: float64
```

2. Variance threshold

```
[4]: from sklearn.feature_selection import VarianceThreshold
     vt=VarianceThreshold(threshold=0.01)
     vt.fit(X_train)

     low_variance_features=X_train.columns[~vt.get_support()]
     low_variance_features
```

```
[4]: Index([], dtype='object')
```

3. VIF (multicollinearity check)

	feature	vif
2	bmi	10.031681
0	age	8.123823
3	children	3.079653
5	exercise_frequency	2.688370
6	coverage_level	2.409913
4	smoker	1.953795
12	med_h_no_record	1.915086
10	med_h_heart disease	1.915067

4. Linear coefficients

```
coef=pd.Series(model.coef_, index=X_train.columns).sort_values(ascending=True)
coef.head()
```

```
fam_med_h_no_record             -867.128325
med_h_no_record                 -865.845489
occ_unemployed                  -642.432681
gender                          -488.444795
fam_med_h_high blood pressure   -433.060098
dtype: float64
```

5. LASSO shrinkage

```
lcoef=pd.Series(lasso.coef_, index=X_train.columns).sort_values(ascending =True)
lcoef
```

```
[47]: fam_med_h_no_record             -866.276265
      med_h_no_record                 -864.985182
      occ_unemployed                  -641.500015
      gender                          -487.945112
      fam_med_h_high blood pressure   -432.170124
      med_h_high blood pressure       -431.738917
      occ_student                     -426.509287
      region_southwest                -343.666968
      region_northwest                -300.538433
      region_southeast                -213.653191
      occ_unknown                     -103.626456
      occ_white collar                 215.646838
      age                              271.113425
      children                         336.090186
      bmi                              461.519182
```

6. Random Forest feature importance

```
[57]: importance=pd.Series(model.feature_importances_, index=X_train.columns).sort_values(ascending=True)
      importance
```

```
[57]: occ_unknown                     0.000614
      med_h_high blood pressure       0.006365
      fam_med_h_high blood pressure   0.006438
      children                        0.006661
      age                             0.007866
      occ_student                     0.008810
      occ_white collar                0.009351
      gender                          0.012150
      occ_unemployed                  0.012419
      bmi                             0.016950
      med_h_no_record                 0.019050
      fam_med_h_no_record             0.019466
      exercise_frequency              0.028074
      fam_med_h_heart disease         0.153693
      med_h_heart disease             0.155016
```

Consensus Finding

Strong predictors:

- Smoker, Coverage level, Medical history, Family medical history, Exercise frequency

Weak but retained:

- Gender, Children, Occupation

Reason: Some models still used them and no multicollinearity risk

Region dropped since consistently weak

Models Selection

Model	R^2	Training Time	Inference	Model Size
Linear Regression	~0.985	**0.31 sec**	**0.01 sec**	**~0.8 KB**
Lasso	~0.985	0.37 sec	0.006 sec	0.75 KB
Random Forest	~0.986	98 sec	8.39 sec	**~GBs**

Hence, Final Model Decision: Lasso (Production-friendly)

- Similar accuracy to Random Forest, 10,000× smaller, Real-time API feasible, Interpretable coefficients, Low infra cost

---

Final Pipeline architecture

1. **Train.py**

- Load Regression.db
- Business Split (700k train / 200k val)
- Calls preprocess on both train and validation data
- Train Final Model -> lasso
- Evaluate (R2, RMSE)
- Save **model.pkl + scaler.pkl +feature_columns.json**

```
(base) C:\Users\shambhavic\OneDrive - Temenos\Data Science and Analytics\jupyteruploads\Capstone Project\Regression\Insurance premium prediction>python trai
n.py
Data loaded: (1000000, 12)
(700000, 12) (200000, 12) (100000, 12)
Validation R2: 0.9870
Validation RMSE: 502.77
Model and scaler saved.
```

2. **Preprocessing.py**

- Copy dataframe
- Handle missing values
- Text standardization (strip + lower)
- Encoding: Binary, Ordinal, One-hot
- Drop region
- Separate target if present
- Scale numeric columns
- Return X, y, scaler

3. **Predict.py**

- Load model.pkl, scaler.pkl and feature_columns.json
- Load production data (last 100k)
- preprocess(prod_df, scaler, training=False)
- Align columns with training schema
- Predict premiums
- Save predicted_premiums.csv

```
Model and scaler saved.

(base) C:\Users\shambhavic\OneDrive - Temenos\Data Science and Analytics\jupyteruploads\Capstone Project\Regression\Insurance premium prediction>python pred
ict.py
Data loaded: (1000000, 12)
Production data shape: (100000, 12)
   predicted_charges
0       17619.217455
1        9935.562123
2       14560.510963
3       16563.611803
4       16359.428511
Predictions saved to predicted_premiums.csv
```

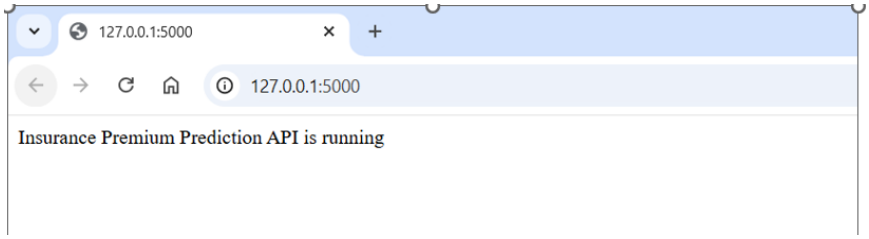
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	gender	bmi	children	smoker	region	medical_history	family_medical_history	exercise_frequency	occupation	coverage_level	charges	predicted_charges		
2	64	male	34.67	2	no	northwest	Heart disease	Heart disease	Never	Student	Basic	17685.37836	17619.21746		
3	18	male	40.19	5	no	southwest	High blood pressure		Rarely	Blue collar	Basic	9547.613014	9935.562123		
4	27	female	46.74	4	no	northeast	High blood pressure	Diabetes	Never	Student	Premium	14692.00132	14560.51096		

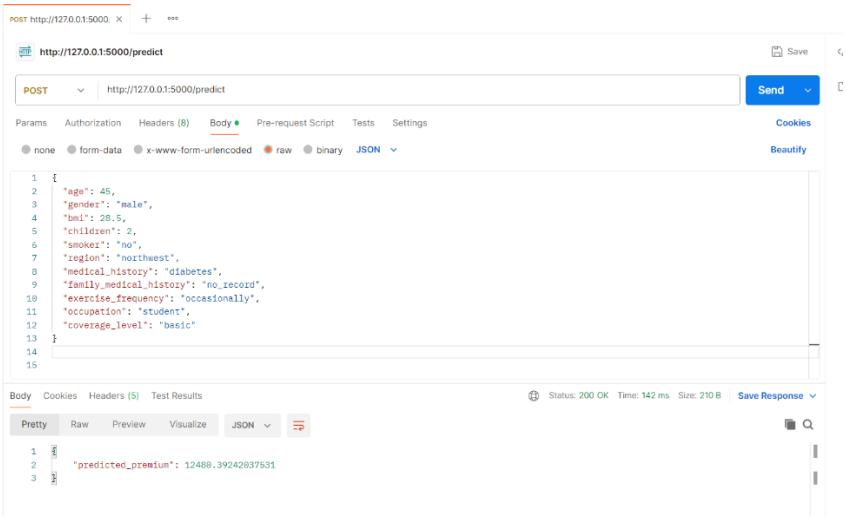
4. **App.py**

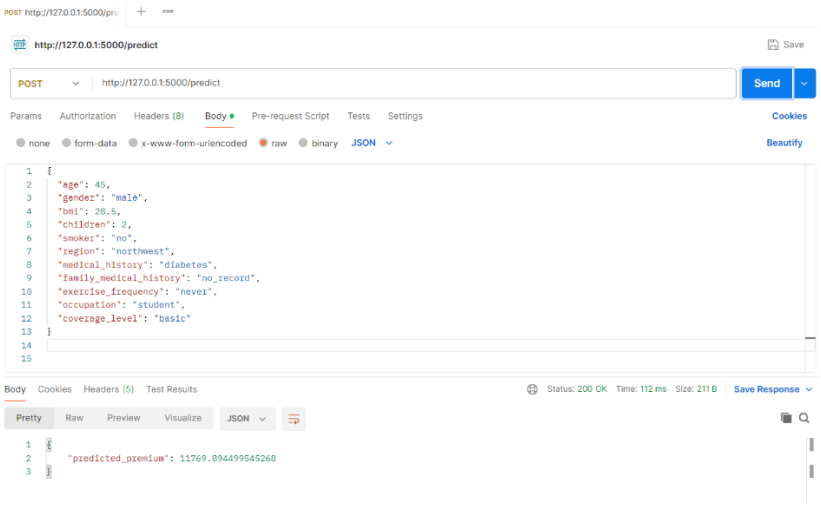
- User JSON inputConvert to DataFrame
- preprocess(input_df, scaler, training=False)
- Align columns
- model.predict()
- Return premium JSON

```
Predictions saved to predicted_premiums.csv

(base) C:\Users\shambhavic\OneDrive - Temenos\Data Science and Analytics\jupyteruploads\Capstone Project\Regression\Insurance premium prediction>python app.
py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 357-612-680
127.0.0.1 - - [08/Feb/2026 18:39:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Feb/2026 18:40:34] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [08/Feb/2026 18:43:01] "POST /predict HTTP/1.1" 200 -
 * Detected change in 'C:\\Users\\shambhavic\\AppData\\Local\\anaconda3\\Scripts\\conda-script.py', reloading
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 357-612-680
```




POST http://127.0.0.1:5000/
http://127.0.0.1:5000/predict
Save
POST
http://127.0.0.1:5000/predict
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"age": 45,
"gender": "male",
"bmi": 28.5,
"children": 2,
"smoker": "no",
"region": "northwest",
"medical_history": "diabetes",
"family_medical_history": "no_record",
"exercise_frequency": "occasionally",
"occupation": "student",
"coverage_level": "basic"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 142 ms
Size: 210 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
"predicted_premium": 12488.39242037531


POST http://127.0.0.1:5000/pre
http://127.0.0.1:5000/predict
Save
POST
http://127.0.0.1:5000/predict
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"age": 45,
"gender": "male",
"bmi": 28.5,
"children": 2,
"smoker": "no",
"region": "northwest",
"medical_history": "diabetes",
"family_medical_history": "no_record",
"exercise_frequency": "never",
"occupation": "student",
"coverage_level": "basic"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 112 ms
Size: 211 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
"predicted_premium": 11769.894499545268