Medical Insurance Cost Prediction Project

Project Overview

This project analyzes the Medical Insurance Dataset and builds predictive models to estimate insurance charges using machine learning techniques. The workflow covers data loading, exploration, preprocessing, visualization, model training, and evaluation.

Dataset Information

The dataset contains 1338 records and 7 features:

- age (int)
- sex (object)
- bmi (float)
- children (int)
- smoker (object)
- region (object)
- charges (float)

All features are non-null and complete.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/insurance.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
df.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
df.shape
```

```
(1338, 7)
```

Exploratory Data Analysis (EDA)

- Summary statistics
- Data types inspection

- Basic visualizations

```
df.isna().sum()
```

	0
age	0
sex	0
bmi	0
children	0
smoker	0
region	0
charges	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(1)
```

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
df_num = df.select_dtypes(include='number')
df_cat = df.select_dtypes(exclude='number')
```

```
for col in df_num.columns:
    plt.figure(figsize=(12, 8))
    sns.boxplot(df_num[col])
    plt.title(col)
    plt.show()
```

```
for col in df_num.columns:
    q1 = df_num[col].quantile(0.25)
    q3 = df_num[col].quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5*iqr
    upper = q1 + 1.5*iqr
    df_num[col] = df_num[col].clip(lower=lower, upper=upper)
```

```
df_encoded = pd.get_dummies(df_cat, columns=df_cat.columns, drop_first=True)
```

```
df = pd.concat([df_num, df_encoded], axis=1)
```

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop('charges', axis=1)
y = df['charges']
```

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

Model Development

Models typically include:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression

Linear Regression

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

Evaluation Metrics

Common metrics used:

- R^2 Score
- MSE
- RMSE

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

```
print(r2)
print(mse)
```

```
0.8090761098707089
10221312.935341042
```

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree=3)
```

```
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.fit_transform(X_test)
```

```
model.fit(X_train_poly, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred_poly = model.predict(X_test_poly)
```

```
r2 = r2_score(y_test, y_pred_poly)
mse = mean_squared_error(y_test, y_pred_poly)
```

```
print(r2)
print(mse)
```

```
0.8013531097587564
10634771.936676513
```

Ridge Regression

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

```
from sklearn.linear_model import Ridge

ridge = Ridge(alpha=10)
ridge.fit(X_train_scaled, y_train)
```

```
▾ Ridge
Ridge(alpha=10)
```

```
y_pred_ridge = ridge.predict(X_test_scaled)
```

```
r2 = r2_score(y_test, y_pred_ridge)
mse = mean_squared_error(y_test, y_pred_ridge)
```

```
print(r2)
print(mse)
print(r2)
```

```
0.8057432870896143
10399739.137428861
0.8057432870896143
```

Lasso Regression

```
from sklearn.linear_model import Lasso
```

```
lasso = Lasso(alpha=1)
lasso.fit(X_train_scaled, y_train)
```

```
▾ Lasso
Lasso(alpha=1)
```

```
y_pred_lasso = lasso.predict(X_test_scaled)
```

```
r2 = r2_score(y_test, y_pred_lasso)
mse = mean_squared_error(y_test, y_pred_lasso)
```

```
print(r2)
print(mse)
```

```
0.8066691928194969
10350169.79224452
```

Summary

Linear Regression provides the best fit with an R^2 of about 0.809. Other models (Ridge, Lasso, Polynomial) show similar performance, indicating the dataset is largely linear and gains little from regularization or polynomial expansion. Summaries of insights and model comparison outcomes.