CSSM 502 – Course Project Report

1. Research Goal and Data

In this project, I will investigate which regressor predicts medical cost of beneficiaries best for an insurance company. I decided to develop regression models for 'insurance_data.csv' data. I can explain my data as following:

- **Age:** age of primary beneficiary,
- **Sex:** insurance beneficiary gender (female, male),
- **BMI:** body mass index, providing an understanding of body weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9,
- **Children:** number of children covered by health insurance / number of dependents,
- **Smoker:** smoking (yes/no),
- **Medical Cost:** individual medical costs billed by health insurance.

2. Data Preprocessing

My input variables are 'age', 'sex', 'bmi', 'children' and 'smoker'. My target variable is 'medical cost'. In 'bmi' column there are some missing values. I replaced them with most frequent item in 'bmi' column. Then I constructed features and target matrices. In features matrix, 'sex', 'smoker' columns have categorical information. Therefore, I used one-hot encoding on features matrix.

```python
#read from csv file
df = pd.read_csv('insurance_data.csv', delimiter=';')
df['bmi'] = df['bmi'].replace(',','.', regex=True).astype(float)
df['medical cost'] = df['medical cost'].replace(',','.', regex=True).astype(float)


#construct features and target matrices
X = df.iloc[:,0:5]
Y = df['medical cost']

#for bmi column there are missing data I replaced missing values with most_frequent it
X_processed = SimpleImputer(strategy='most_frequent', missing_values=99)
X_processed = X_processed.fit(X[['bmi']])
X['bmi'] = X_processed.transform(X[['bmi']])

# sex and smoker data are categorical, thefore I will apply one-hot encoding
# creating instance of one-hot-encoder
X_encoded = OneHotEncoder(sparse=True, handle_unknown='ignore')
# passing sex and smoker columns
X_encoded = pd.DataFrame(X_encoded.fit_transform(X[['sex', 'smoker']]).toarray())
# merge encoded columns with other columns
X_encoded = X_encoded.join(X['age'])
X_encoded = X_encoded.join(X['bmi'])
X_encoded = X_encoded.join(X['children'])
```

3. Model Development

I split existing data as train (%80) data and test (default %20) data.

```
#split the data as train and test
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X_encoded, Y, random_state=0,
train_size=0.8, test_size=0.2)
```

First, I used LinearRegression model as regressor. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
#LinearRegression
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True) #storing of hyperparameter values
model.fit(Xtrain, Ytrain)
y_pred_LR = model.predict(Xtest)
#calculated MSE to measure model performance
MSE_LR = mean_squared_error(Ytest, y_pred_LR)
MAE_LR = mean_absolute_error(Ytest, y_pred_LR)
MSE_LR,MAE_LR
```

```
(32198271.422296938, 3936.2671465688218)
```

I decided to optimize LinearRegression model. For optimization, I used GridSearchCV module. By help of this module, I tried following hyperparameters:

- **fit_intercept**: [True, False],
- **normalize**: [True, False],
- **copy_X**: [True, False]

GridSearchCV module returned best parameters as following:

- **fit_intercept**: [True],
- **normalize**: [True],
- **copy_X**: [True]

```
#For LR optimization, I will use GridSearchCV
from sklearn.model_selection import GridSearchCV
import numpy as np

grid_params = {'fit_intercept': [True, False],
               'normalize' : [True, False],
               'copy_X' : [True, False]}

grid = GridSearchCV(LinearRegression(), grid_params, verbose = 1, cv = 7, n_jobs = -1)

grid.fit(Xtrain,Ytrain)
grid.best_params_
```

```
Fitting 7 folds for each of 8 candidates, totalling 56 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  56 out of  56 | elapsed:    1.7s finished
{'copy_X': True, 'fit_intercept': True, 'normalize': True}
```

I run LinearRegression with best hyperparameter values. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
#run LinearRegression with optimized parameters
model_optimized = LinearRegression(fit_intercept=True, normalize=True,copy_X=True) #st
model_optimized.fit(Xtrain, Ytrain)
y_pred_LR_optimized = model_optimized.predict(Xtest)
#calculated MSE to measure model performance
MSE_LR_optimized = mean_squared_error(Ytest, y_pred_LR_optimized)
MAE_LR_optimized = mean_absolute_error(Ytest, y_pred_LR_optimized)
MSE_LR_optimized,MAE_LR_optimized
```

```
(32198271.422296952, 3936.2671465688272)
```

Then I used SVR as regressor. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
#Support Vector Regressor(SVR)
regressor = SVR(kernel = 'poly')
regressor.fit(Xtrain, Ytrain)
y_pred = regressor.predict(Xtest)
#calculated MSE to measure model performance
MSE = mean_squared_error(Ytest, y_pred)
MAE = mean_absolute_error(Ytest, y_pred)
MSE,MAE
```

```
(171644228.42779985, 8150.237918030387)
```

I decided to optimize SVR model. For optimization, I used GridSearchCV module. By help of this module, I tried following hyperparameters:

- **kernel:** ['linear', 'poly', 'rbf', 'sigmoid']
- **gamma**: ['scale', 'auto']

GridSearchCV module returned best parameters as following:

- **kernel:** ['poly']
- **gamma**: ['auto']

```
#For SVR optimization, I will use GridSearchCV
from sklearn.model_selection import GridSearchCV
import numpy as np

grid_params = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
               'gamma' : ['scale', 'auto']}

grid = GridSearchCV(SVR(), grid_params, verbose = 1, cv = 7, n_jobs = -1)

grid.fit(Xtrain,Ytrain)
grid.best_params_
```

```
Fitting 7 folds for each of 8 candidates, totalling 56 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done  56 out of  56 | elapsed:  1.7min finished
```

```
{'gamma': 'auto', 'kernel': 'poly'}
```

I run SVR with best hyperparameter values. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
#run SVR with best parameters
regressor_optimized = SVR(kernel = 'poly', gamma = 'auto')
regressor_optimized.fit(Xtrain, Ytrain)
y_pred_optimized = regressor_optimized.predict(Xtest)
MSE_optimized = mean_squared_error(Ytest, y_pred_optimized)
MAE_optimized = mean_absolute_error(Ytest, y_pred_optimized)
MSE_optimized,MAE_optimized
```

```
(21778183.016043536, 2141.3520245314526)
```

Finally, I used Random Forest Regressor(RFR) as regressor. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
from sklearn.ensemble import RandomForestRegressor
#Random Forest Regressor(RFR)
regressor_RFR = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor_RFR.fit(Xtrain, Ytrain)
y_pred_RFR = regressor_RFR.predict(Xtest)
#calculated MSE to measure model performance
MSE_RFR = mean_squared_error(Ytest, y_pred_RFR)
MAE_RFR = mean_absolute_error(Ytest, y_pred_RFR)
MSE_RFR,MAE_RFR
```

```
(20354599.438501805, 2588.707771231343)
```

I decided to optimize RFR model. For optimization, I used GridSearchCV module. By help of this module, I tried following hyperparameters:

- **criterion:** ['mse', 'mae']
- **max_features**: ['auto', 'sqrt', 'log2']
- **bootstrap:** [True, False]

GridSearchCV module returned best parameters as following:

- **criterion:** ['mae']
- **max_features**: ['sqrt']
- **bootstrap:** [True]

```
#For RFR optimization, I will use GridSearchCV
grid_params = {'criterion' : ['mse', 'mae'],
               'max_features' : ['auto', 'sqrt', 'log2'],
               'bootstrap' :[True, False]}

max_features : {"auto", "sqrt", "log2"}

grid = GridSearchCV(RandomForestRegressor(), grid_params, verbose = 1, cv = 7, n_jobs

grid.fit(Xtrain,Ytrain)
grid.best_params_
```

```
Fitting 7 folds for each of 12 candidates, totalling 84 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    3.7s
[Parallel(n_jobs=-1)]: Done  84 out of  84 | elapsed:   11.4s finished
```

```
{'bootstrap': True, 'criterion': 'mae', 'max_features': 'sqrt'}
```

I run RFR with best hyperparameter values. I fitted the model, predicted values for Xtest, calculated accuracy of model with MSE (Mean Squared Error) and MAE (Mean Absolute Error).

```
#run RFR with best parameters
regressor_RFR_optimized = RandomForestRegressor(n_estimators = 100, random_state = 0,
regressor_RFR_optimized.fit(Xtrain, Ytrain)
y_pred_RFR_optimized = regressor_RFR_optimized.predict(Xtest)
#calculated MSE to measure model performance
MSE_RFR_optimized = mean_squared_error(Ytest, y_pred_RFR_optimized)
MAE_RFR_optimized = mean_absolute_error(Ytest, y_pred_RFR_optimized)
MSE_RFR_optimized,MAE_RFR_optimized
```

```
(16988783.590700284, 2285.703164365672)
```

4. Result

I summarized performances of regressors in following table. My metrics are MSE (Mean Squared Error) and MAE (Mean Absolute Error). According to these metrics, SVR and RFR are better than Linear Regression. I will choose RFR as a regressor for this data.

Regressor	**MSE**	**MAE**
Linear Regression	32.198.271,42	3.936,26
Support Vector Regressor	21.778.183,01	2.141,35
Random Forest Regressor	16.988.783,59	2.285,70