به نام خدا

گزارش تمرین دوم – مبانی علم داده **نیایش خانی ۹۹۵۲۱۲۳۵**

مرحله ۱: Missing Values

اولین کار هندل کردن مقادیر از دست رفته یا missing values است که یا باید هر فیچر با مقادیر miss شده را حذف کرد و یا با مقدار مناسب آن را پر کرد. از آنجایی که قبلا (در پارت ۱) ویژگی‌هایی که بیش از ۵۰ درصد مقادیر از دست رفته داشتند را حذف کردیم، این بار تصمیم گرفتم هر آنها را پر کنم. از بین روش هایی که برای این کار وجود دارد مانند جایگزین کردن مقادیر با میانه یا میانگین ستون، با سرچ به این نتیجه رسیدم که می‌توان مقادیر از دست رفته را پیش‌بینی و آنها را پر کرد.

```
# Define the regressor with early stopping
regressor = GradientBoostingRegressor(n_estimators=100, validation_fraction=0.1,
                                      n_iter_no_change=10, tol=0.01)

# Create the iterative imputer using the regressor
imputer = IterativeImputer(estimator=regressor)

# Train on the training features
imputer.fit(train_features)

# Transform both training data and testing data
X = imputer.transform(train_features)
X_test = imputer.transform(test_features)
```

بنابراین این کار را با استفاده از یک مدل Gradient Boosting Regression انجام دادم. این روش با استفاده از imputation های iterative، روابط چند متغیره بین فیچرها را مدیریت می‌کند و در مقایسه با روش‌های ساده‌تر مثل میانگین یا میانه، انتساب‌های قوی‌تری تولید می‌کند. از پارامترهای early stopping هم برای جلوگیری از overfit استفاده کردم.

همانطور که مشاهده می‌شود با انجام این کار بر روی داده‌های train و تست، دیگر مقدار از دست رفته ای وجود ندارد.

مرحله ۲: Scaling Features

مرحله بعدی اسکیل کردن است زیرا فیچرها اغلب دارای واحدهای مختلفی هستند و نرمالیزه کردن آنها تضمین می‌کند که واحدها بر عملکرد الگوریتم‌هایی مثل SVM و K-nearest تأثیر نمی گذارند. برای این کار از MinMaxScaler استفاده می‌کنیم تا مقادیر را بین صفر و یک اسکیل کند.

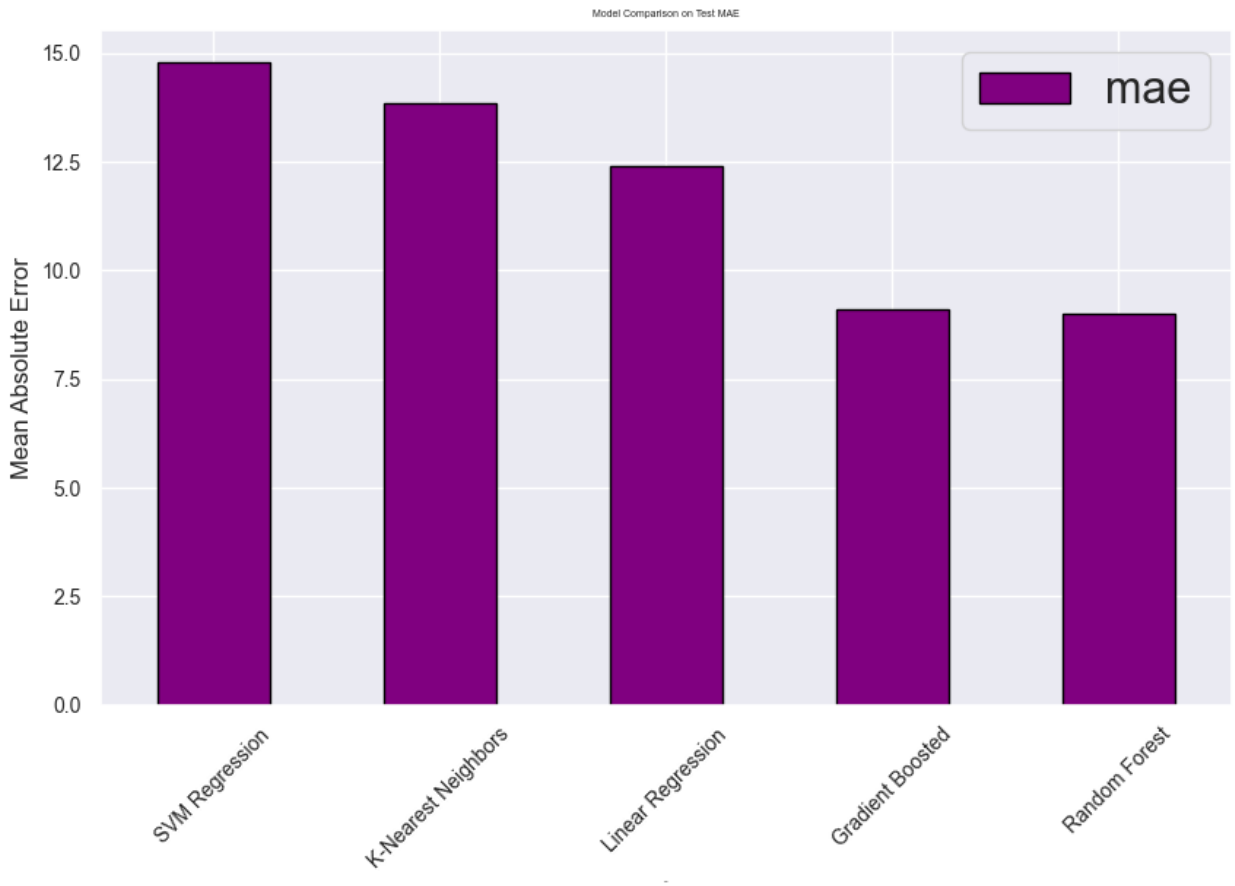
مرحله ۳: Use Different Models

حال به سراغ آموزش دادن مدل‌های مختلف می‌رویم. مراحل زیر را برای انواع مدل‌ها انجام می‌دهیم و نتایج را با پارامتر MAE مقایسه می‌کنیم. (در کدهای موجود پارامتر MSE نیز در نظر گرفته شده زیرا در ابتدای نوت‌بوک ذکر شده بود.)

مراحل آموزش مدل :

۱. تعریف مدل، ۲. آموزش دادن آن، ۳. پیش‌بینی بر روی داده تست، ۴. محاسبه MSE و MAE

این مراحل برای مدل‌های SVM، Linear، Random Forest، K-Nearest Neighbors و Gradient Boosting انجام شده است و نتایج آن با استفاده از نمودار زیر قابل مقایسه است.



همانطور که مشاهده می‌شود این مدل‌ها بر اساس مقدار MAE مرتب شده‌اند. مدل SVM و K-Nearest بدترین عملکرد و Gradient Boost و Random Forest بهترین عملکرد را دارند. البته این دو متد بسیار نزدیک به هم هستند و نمی‌توان بهترین را انتخاب کرد.

مرحله ۴: Hyperparameter Tuning

Underfitting زمانی رخ می دهد که مدل برای ثبت الگوهای داده ها بسیار ساده باشد که با افزایش پیچیدگی مدل، این مشکل حل می‌شود. Overfitting نیز زمانی اتفاق می افتد که مدل داده های آموزشی را به خاطر بسپارد که این نیز با کاهش پیچیدگی یا به regularization برطرف می‌شود. فایده تنظیم هایپرپارامترها این است که با متعادل کردن آندرفیت و اورفیت، بر عملکرد و performance یک مدل تأثیر می‌گذارد و آن را بهبود می‌بخشد. مشکل انتخاب هایپرپارامترها این است که هیچ مجموعه ای در تمام مشکلات بهترین کار را نخواهد داشت. بنابراین، برای هر مجموعه داده جدید، ما باید بهترین تنظیمات را پیدا کنیم.

من این کار را از طریق جستجوی تصادفی و cross validation متقاطع انجام دادم.

در جستجوی تصادفی ما هایپراپارامترها را برای ارزیابی انتخاب می‌کنیم، طیف وسیعی از گزینه‌ها را تعریف می‌کنیم و سپس به‌طور تصادفی ترکیب‌هایی را برای امتحان انتخاب می‌کنیم. Cross validation هم برای ارزیابی عملکرد هایپرپارامترها استفاده می شود. در اینجا ما جستجوی تصادفی را با cross validation برای انتخاب هایپرپارامترهای بهینه برای رندوم فارست که بهترین عملکرد را در مرحله قبل داشت، با استفاده از Scikit-Learn و RandomizedSearchCV انجام خوهیم داد.

در این کار مدل RandomForestRegressor را با دیتاست X و y هنگام تست کردن ترکیب‌های متفاوتی از هایپرپارامترها که در param_grid تعریف شده است، آموزش می‌دهیم. best_estimator_ بهترین ترکیب هایپرپارامتر را که پیدا کرده است ریترن می‌کند.

```
random_cv.best_estimator_
✓ 0.0s

RandomForestRegressor
RandomForestRegressor(max_depth=7, min_samples_leaf=5, min_samples_split=3,
                      n_estimators=678)
```

best_score_ هم بهترین امتیازی که مدل با استفاده از هایپرپارامترهای بهینه بدست آورده است را برمی‌گرداند.

```
random_cv.best_score_
✓ 0.0s

0.801642460447551
```

سپس این مدل را با استفاده از هایپرپارامترهای بدست آمده آموزش می‌دهیم، پیش‌بینی کرده و سپس عملکرد مدل را ارزیابی می‌کنیم. MAE بدست آمده برای این مدل ۹/۱۵ است.

در آخر مدل به دست آمده را روی داده تست استفاده می‌کنیم. فاینال مدل، مدل نهایی ما خواهد بود که MAE آن **9.16** ثبت شده‌است.

```
final_model.fit(X, y)

final_pred = final_model.predict(X_test)

print('Final model MAE = %0.4f' % mae(y_test, final_pred))
✓ 59.7s

Final model MAE = 9.1601
```

برای اینکه درک بهتری از نحوه عملکرد این مدل داشته باشیم، نموداری از مقادیر درست در دیتای تست و مقادیر پیش‌بینی شده را رسم کرده ام.

همانطور که مشاهده می‌شود توزیع این دو تقریبا نزدیک به هم هستند و می‌توان نتیجه گرفت که مدل عملکرد خوب و قابل قبولی دارد.