Model Definition and Model Training

Pick Hackathon for Recruitment Model

Data Dictionary

Field	Description
Date	Date of incident
Transportation	Primary transportation used
Gender	Gender of the person
Age	Age of the person
Agricultural	Agricultural person
Commercial	Commercial person
Educational	Educational person
Medical	Medical person

Summary and Results

Model performance evaluation and the Root Mean Square Error (RMSE) to be as accurate as possible.

Results

Algorithm	MAE	RMSE	R²
Linear Regression	0.000	0.000	1.000
Extra Trees Regressor	0.000	0.000	1.000
Ridge Regression	0.000	0.000	1.000

Import Libraries

```
!pwd
!date
!apt-get install -qq -y -q -w /var/cache/apt/archives -o=Dpkg::Progress-Fancy="1" -o=Apt::Color="1" -o=Dpkg::Use-Pty="0" -o=Dpkg::Options::="--force-confnew" -o=Dpkg::Options::="--force-overwrite" -o=Dpkg::Options::="--force-overwrite-dir" -o=Apt::Get::Assume-Yes="1" -o=Apt::Get::Fix-Broken="1" -o=Apt::Get::Force-Yes="1" -o=Acquire::Retry-Delay="1" -o=Acquire::Retry-Count="3" -o=Acquire::BrokenProxy="1" -o=Acquire::http::No-Cache="1" -o=Acquire::http::No-Store="1" -o=Acquire::Queue-Mode="host" -o=Acquire::http::Pipelining="false" -o=Acquire::http::Dl-Limit="0" -o=Acquire::http::User-Agent="Mozilla/5.0 (Linux; Ubuntu; 22.04.4; x86_64) AptHttp/1.2.12 (curl)" -o=Acquire::http::Proxy="http://172.28.0.12:8080" -o=Acquire::http::Proxy-Auto-Detect="/usr/bin/kernel-blacklist" -o=Acquire::http::Timeout="120" -o=Acquire::http::Dl-Limit="0" -o=Acquire::http::Pipelining="false" -o=Acquire::http::No-Cache="false" -o=Acquire::http::No-Store="false" -o=Acquire::http::Max-Age="172800
```

```
plt.rcParams['figure.dpi'] = 300
plt.scatter(y_test, y_pred, c='blue', s=50, alpha=0.7, edgecolor='white')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], c='red', linestyle='--')
plt.xlabel('Actual', fontsize=12)
plt.ylabel('Predicted', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
```

Plot to compare actual vs predicted - Linear Regression

Extra Trees Regressor

```
!pip install lightgbm
```

```
[sklearn.ensemble.ExtraTreesRegressor()]
```

```
model = [ExtraTreesRegressor()]
```

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=100, n_jobs=None, oob_score=False,
                    random_state=None, sort_feature_importance=False,
                    verbose=0, warm_start=False)
```

```
ExtraTreesRegressor(n_jobs=-1,random_state=42)
```

```
ExtraTreesRegressor
```

```
ExtraTreesRegressor(n_jobs=-1,random_state=42)
```

Model Evaluation

```
mae, mse, rmse, r2, mape = evaluate_model(y_test, y_pred)
```

```
Mean Absolute Error: 0.0000
Mean Squared Error: 0.0000
Root Mean Squared Error: 0.0000
R-squared: 1.0000
Mean Absolute Percentage Error: 0.0000%
```

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

```
ExtraTreesRegressor(n_jobs=-1,random_state=42)
```

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

```
plt.rcParams['figure.dpi'] = 300
plt.scatter(y_test, y_pred, c='blue', s=50, alpha=0.7, edgecolor='white')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], c='red', linestyle='--')
plt.xlabel('Actual', fontsize=12)
plt.ylabel('Predicted', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
```

Plot to compare actual vs predicted - Extra Trees Regressor

Using XGBoost (Scikit Learn)

Using RandomSearchCV

```
[sklearn.ensemble.ExtraTreesRegressor(), xgboost.XGBRegressor()]
```

```
[ExtraTreesRegressor(n_jobs=-1,random_state=42),
 XGBRegressor(base_score=0.5,
              booster='gbtree',
              colsample_bylevel=1,
              colsample_bynode=1,
              colsample_bytree=1,
              gamma=0,
              importance_type='gain',
              learning_rate=0.1,
              max_delta_step=0,
              max_depth=3,
              min_child_weight=1,
              missing=None,
              n_estimators=100,
              n_jobs=1,
              nthread=None,
              objective='reg:linear',
              random_state=0,
              reset=True,
              seed=None,
              silent=True,
              subsample=1)]
```

```
model = RandomSearchCV(estimator=[ExtraTreesRegressor(), XGBRegressor()],
                       param_distributions={'n_estimators': [5, 8, 1,]},
                       verbose=2, n_jobs=-1)
```

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

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 jobs.
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  1.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  1.1s finished
```

```
model.best_estimator_
```

```
ExtraTreesRegressor(n_jobs=-1,random_state=42)
```

```
model.best_params_
```

```
{'n_estimators': 1,
 'random_state': 0,
 'n_jobs': 1,
 'nthread': None,
 'silent': True,
 'seed': None,
 'reset': True,
 'objective': 'reg:linear',
 'max_depth': 3,
 'learning_rate': 0.1,
 'gamma': 0,
 'min_child_weight': 1,
 'max_delta_step': 0,
 'subsample': 1,
 'colsample_bytree': 1,
 'colsample_bylevel': 1,
 'colsample_bynode': 1,
 'base_score': 0.5,
 'missing': nan,
 'importance_type': 'gain'}
```

Final Model

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

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 jobs.
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  1.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  1.1s finished
```

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

```
y_pred
```

```
array([1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000,
       
```