In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
import random
from sklearn.metrics import confusion_matrix
```

In [ ]:
```python
# Load data
df = pd.read_csv('final_dataset.csv', usecols=['CrashType', 'DrvrAlcFlg', 'DrvrVehTyp', 'LightCond', 'NumLanes', 'PedAg
                                                    'PedAlcFlag', 'PedPos', 'PedSex', 'RdCharacte', 'SpeedLimit', 'T
```

In [3]:
```python
df.columns
```

Out[3]:
```
Index(['PedInjury', 'CrashType', 'DrvrAlcFlg', 'DrvrVehTyp', 'LightCond',
       'NumLanes', 'PedAgeGrp', 'PedAlcFlag', 'PedPos', 'PedSex', 'RdCharacte',
       'SpeedLimit', 'TraffCntrl'],
      dtype='object')
```

In [4]:
```python
df['PedInjury'].value_counts()
```

Out[4]:

PedInjury	count
B: Suspected Minor Injury	3767
C: Possible Injury	2915
A: Suspected Serious Injury	1390
K: Killed	1144
O: No Injury	473

dtype: int64

In [5]:
```python
injury_map = {
    'O: No Injury': 0,
    'C: Possible Injury': 1,
    'B: Suspected Minor Injury': 2,
    'A: Suspected Serious Injury': 3,
    'K: Killed': 4
}

df['PedInjuryLabel'] = df['PedInjury'].map(injury_map)
```

In [6]:
```
df["PedInjuryLabel"].value_counts()
```

Out[6]:

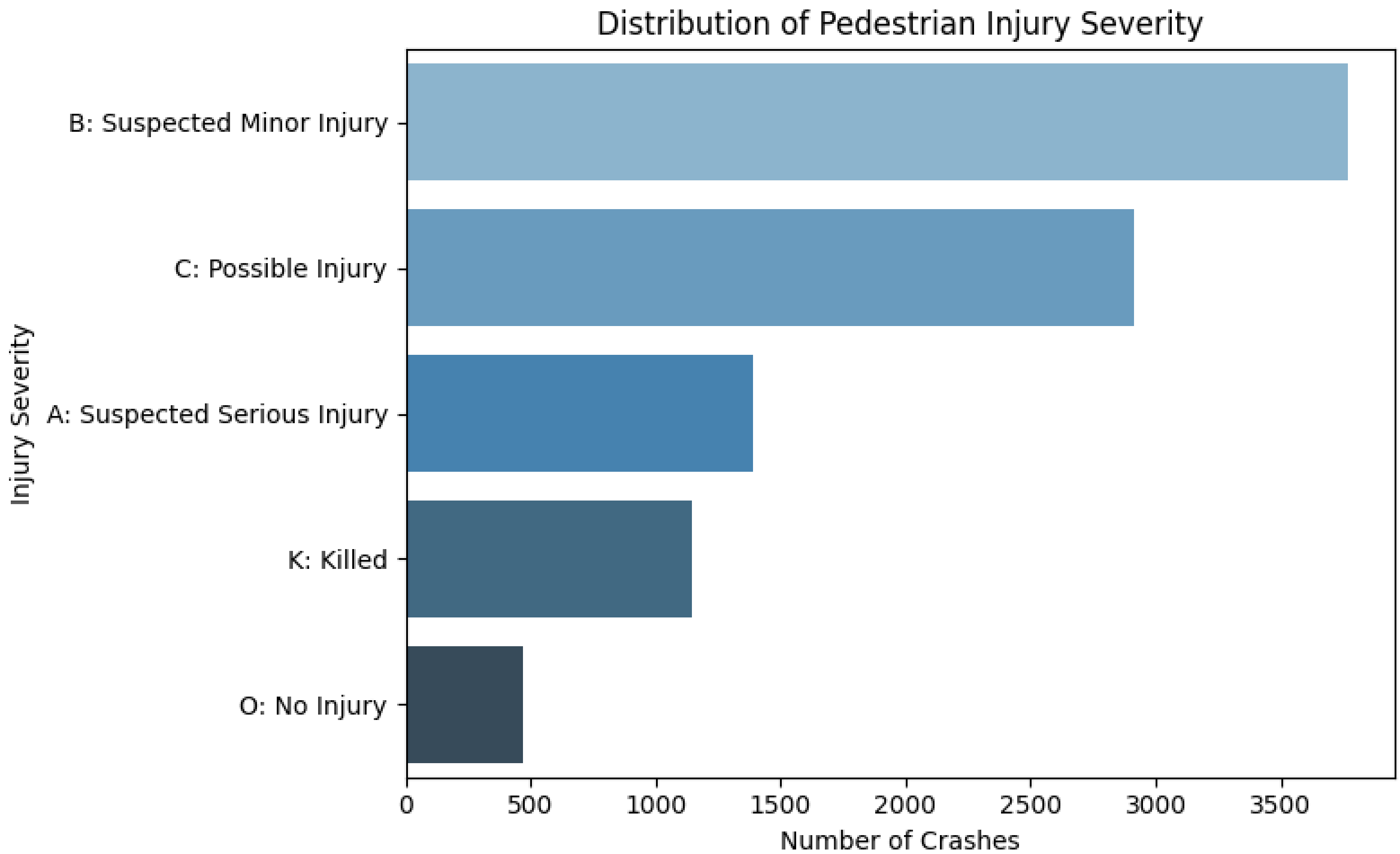
PedInjuryLabel	count
2	3767
1	2915
3	1390
4	1144
0	473

dtype: int64

In [7]:
```
df['BinarySevere'] = df['PedInjuryLabel'].apply(lambda x: 1 if x >= 3 else 0)
```

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
sns.countplot(y='PedInjury', data=df, order=df['PedInjury'].value_counts().index, palette='Blues_d')
plt.title('Distribution of Pedestrian Injury Severity')
plt.xlabel('Number of Crashes')
plt.ylabel('Injury Severity')
plt.tight_layout()
plt.savefig('/content/injury_severity_distribution.png', dpi=300)
plt.show()
```

```
<ipython-input-8-d3e4236d9f2a>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
` and set `legend=False` for the same effect.

  sns.countplot(y='PedInjury', data=df, order=df['PedInjury'].value_counts().index, palette='Blues_d')
```



In [9]:
```
features = [
    'CrashType', 'DrvrAlcFlg', 'DrvrVehTyp', 'LightCond',
    'NumLanes', 'PedAgeGrp', 'PedAlcFlag', 'PedPos',
    'PedSex', 'RdCharacte', 'SpeedLimit', 'TraffCntrl'
]

X = df[features].copy()
y = df['BinarySevere']

# Apply one-hot encoding to categorical features
X_encoded = pd.get_dummies(X, drop_first=False).astype(int)
```

In [10]:
```
y.value_counts()
```

Out[10]:

BinarySevere	count
0	7155
1	2534

dtype: int64

In [11]:

```
X = df[features].copy()
y = df['BinarySevere']
X_encoded = pd.get_dummies(X, drop_first=False).astype(int)
```

In [12]:

```
from sklearn.model_selection import train_test_split

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

In [13]:

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
```

In [14]:

```
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import GridSearchCV
```

In [15]:

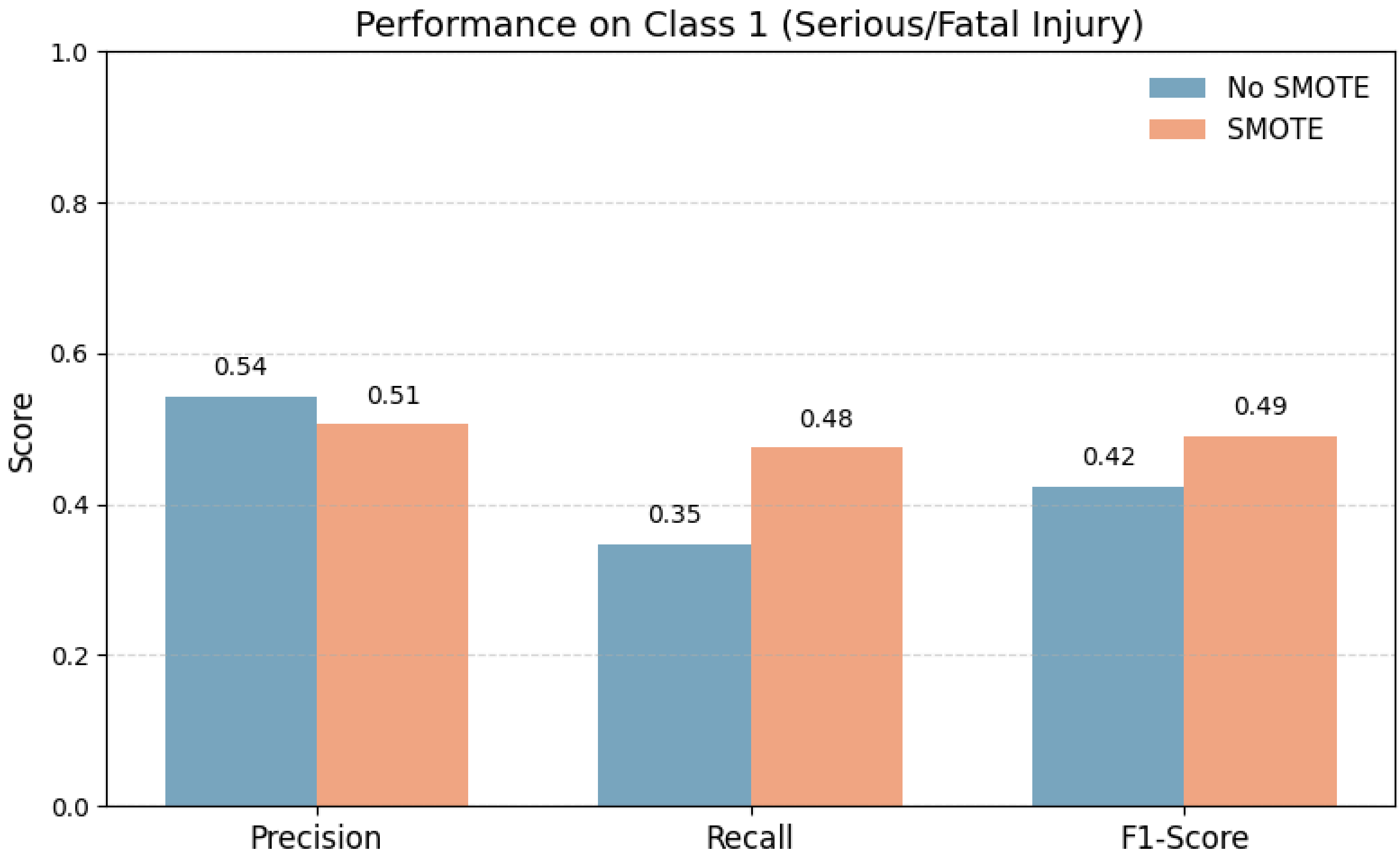
```
#without oversampling
rf_no_smote = RandomForestClassifier(n_estimators=100, random_state=42)
rf_no_smote.fit(X_train, y_train)
y_pred_no_smote = rf_no_smote.predict(X_test)
#baseline with oversampling
rf_smote = RandomForestClassifier(n_estimators=100, random_state=42)
rf_smote.fit(X_train_res, y_train_res)
y_pred_smote = rf_smote.predict(X_test)

report_no_smote = classification_report(y_test, y_pred_no_smote, output_dict=True)
report_smote = classification_report(y_test, y_pred_smote, output_dict=True)
```

```python
compare = pd.DataFrame({
    'No SMOTE': pd.Series(report_no_smote['1']),
    'SMOTE': pd.Series(report_smote['1'])
}).round(3)

print(compare)
```

```
           No SMOTE    SMOTE
precision     0.543    0.506
recall        0.347    0.475
f1-score      0.424    0.490
support     507.000  507.000
```

In [16]:

```python
import matplotlib.pyplot as plt

metrics = ['Precision', 'Recall', 'F1-Score']
values_no_smote = [report_no_smote['1'][m.lower()] for m in metrics]
values_smote = [report_smote['1'][m.lower()] for m in metrics]

x = range(len(metrics))
bar_width = 0.35

plt.figure(figsize=(8, 5))
bars1 = plt.bar(x, values_no_smote, width=bar_width, label='No SMOTE', color='#7aa6c2')
bars2 = plt.bar([i + bar_width for i in x], values_smote, width=bar_width, label='SMOTE', color='#f4a582')


for i in x:
    plt.text(i, values_no_smote[i] + 0.02, f"{values_no_smote[i]:.2f}", ha='center', va='bottom', fontsize=10)
    plt.text(i + bar_width, values_smote[i] + 0.02, f"{values_smote[i]:.2f}", ha='center', va='bottom', fontsize=10)

plt.xticks([i + bar_width / 2 for i in x], metrics, fontsize=12)
plt.ylabel('Score', fontsize=12)
plt.title('Performance on Class 1 (Serious/Fatal Injury)', fontsize=14)
plt.ylim(0, 1)
plt.legend(frameon=False, fontsize=11)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig('/content/smote_effect.png', dpi=300)
plt.show()
```



In [17]:
```
def evaluate(model, name):
    y_pred = model.predict(X_test)
    print(f"\n==== {name} ====")
    print(classification_report(y_test, y_pred))
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
```

In [18]:
```
# --- Random Forest ---
param_rf = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 10, 15],
    'min_samples_split': [2, 5],
}
rf_search = GridSearchCV(RandomForestClassifier(random_state=42), param_rf,
                         scoring='recall', cv=5, n_jobs=-1)
rf_search.fit(X_train_res, y_train_res)
rf_best = rf_search.best_estimator_
```

In [19]:
```
evaluate(rf_best, "Random Forest")
```

```
==== Random Forest ====
              precision    recall  f1-score   support

           0       0.85      0.80      0.82      1431
           1       0.51      0.59      0.55       507

    accuracy                           0.75      1938
   macro avg       0.68      0.69      0.69      1938
weighted avg       0.76      0.75      0.75      1938

Confusion Matrix:
[[1149  282]
 [ 210  297]]
```

In [20]:
```
rf_search.best_params_
```

Out[20]:
```
{'max_depth': 15, 'min_samples_split': 2, 'n_estimators': 200}
```

In [21]:
```
# --- XGBoost ---
param_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3,5,7],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.3, 0.5, 0.7]
}
xgb_search = GridSearchCV(XGBClassifier(objective='binary:logistic',
                                              eval_metric='logloss',
                                              use_label_encoder=False,
                                              random_state=42),
                                param_xgb, scoring='recall', cv=5, n_jobs=-1)
xgb_search.fit(X_train_res, y_train_res)
xgb_best = xgb_search.best_estimator_
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [18:37:25] WARNING: /workspace/src/learner.c
c:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

In [22]:
```
evaluate(xgb_best, "Xgboost")
```

```
==== Xgboost ====
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1431
           1       0.51      0.58      0.54       507

    accuracy                           0.74      1938
   macro avg       0.68      0.69      0.68      1938
weighted avg       0.76      0.74      0.75      1938

Confusion Matrix:
[[1146  285]
 [ 212  295]]
```

In [23]:
```python
xgb_search.best_params_
```

Out[23]:
```
{'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 100, 'subsample': 0.7}
```

In [ ]:
```python
# --- MLP (Neural Network) ---
param_mlp = {
    'hidden_layer_sizes': [(64,), (64, 32), (128, 64)],
    'activation': ['relu', 'tanh'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate_init': [0.001, 0.01]
}
mlp_search = GridSearchCV(MLPClassifier(max_iter=200, random_state=42), param_mlp,
                                scoring='recall', cv=5, n_jobs=-1)
mlp_search.fit(X_train_res, y_train_res)
mlp_best = mlp_search.best_estimator_
```

In [ ]:
```python
evaluate(mlp_best, "MLP")
```

```
==== MLP ====
              precision    recall  f1-score   support

           0       0.80      0.81      0.81      1431
           1       0.45      0.44      0.45       507

    accuracy                           0.71      1938
   macro avg       0.63      0.62      0.63      1938
weighted avg       0.71      0.71      0.71      1938

Confusion Matrix:
[[1156  275]
 [ 283  224]]
```

In [ ]:
```python
mlp_search.best_params_
```

Out[ ]:
```
{'activation': 'relu',
 'alpha': 0.0001,
 'hidden_layer_sizes': (128, 64),
 'learning_rate_init': 0.001}
```

In [24]:
```
# --- Decision Tree ---
param_dt = {
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}
dt_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_dt,
                               scoring='recall', cv=5, n_jobs=-1)
dt_search.fit(X_train_res, y_train_res)
dt_best = dt_search.best_estimator_
```

In [25]:
```
evaluate(dt_best, "Decision Tree")
```

```
==== Decision Tree ====
              precision    recall  f1-score   support

           0       0.86      0.71      0.78      1431
           1       0.45      0.67      0.54       507

    accuracy                           0.70      1938
   macro avg       0.65      0.69      0.66      1938
weighted avg       0.75      0.70      0.72      1938

Confusion Matrix:
[[1018  413]
 [ 168  339]]
```

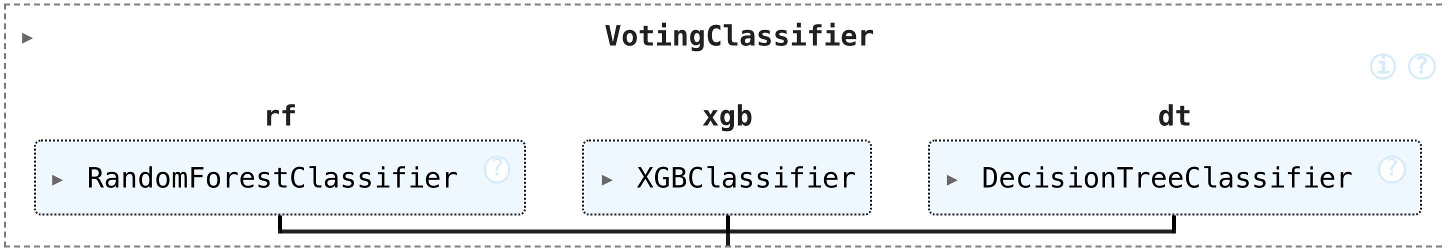
In [26]:
```
dt_search.best_params_
```

Out[26]:
```
{'criterion': 'gini',
 'max_depth': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 10}
```

In [27]:
```
ensemble = VotingClassifier(
    estimators=[
        ('rf', rf_best),
        ('xgb', xgb_best),
        #('mlp', mlp_best),
        ('dt', dt_best)
    ],
```

```python
    voting='soft',
    n_jobs=-1
)
ensemble.fit(X_train_res, y_train_res)
```

Out[27]:



In [36]:

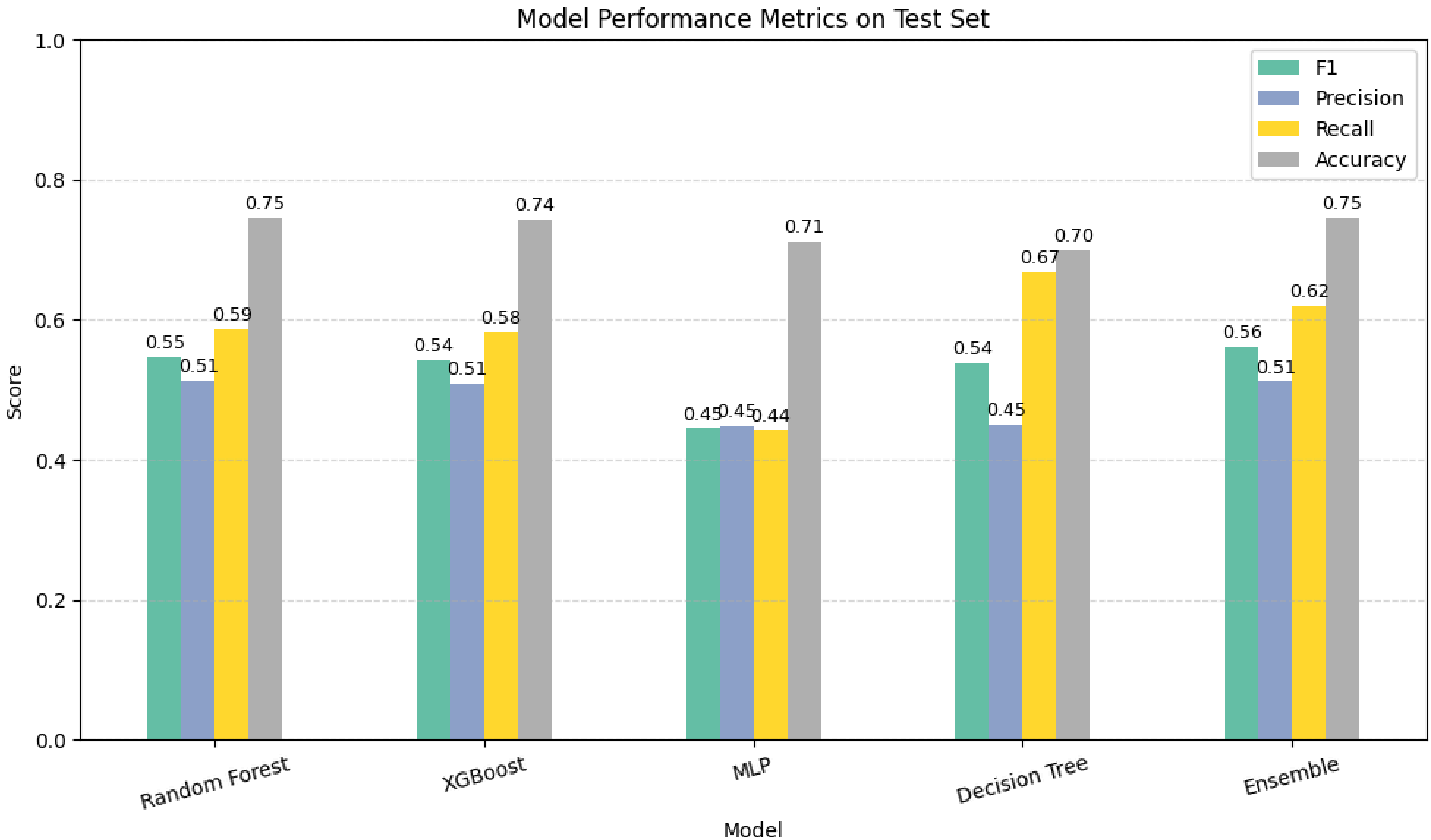
```python
evaluate(ensemble, "Ensemble")
```

```
==== Ensemble ====
              precision    recall  f1-score   support

           0       0.85      0.79      0.82      1431
           1       0.51      0.62      0.56       507

    accuracy                           0.75      1938
   macro avg       0.68      0.71      0.69      1938
weighted avg       0.76      0.75      0.75      1938

Confusion Matrix:
[[1132  299]
 [ 193  314]]
```

In [ ]:

```python
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score
import matplotlib.pyplot as plt
models = {
    'Random Forest': rf_best,
    'XGBoost': xgb_best,
    'MLP': mlp_best,
    'Decision Tree': dt_best,
    'Ensemble': ensemble
}

metrics = {'Model': [], 'F1': [], 'Precision': [], 'Recall': [], 'Accuracy': []}

for name, model in models.items():
    y_pred = model.predict(X_test)
    metrics['Model'].append(name)
    metrics['F1'].append(f1_score(y_test, y_pred))
    metrics['Precision'].append(precision_score(y_test, y_pred))
    metrics['Recall'].append(recall_score(y_test, y_pred))
```

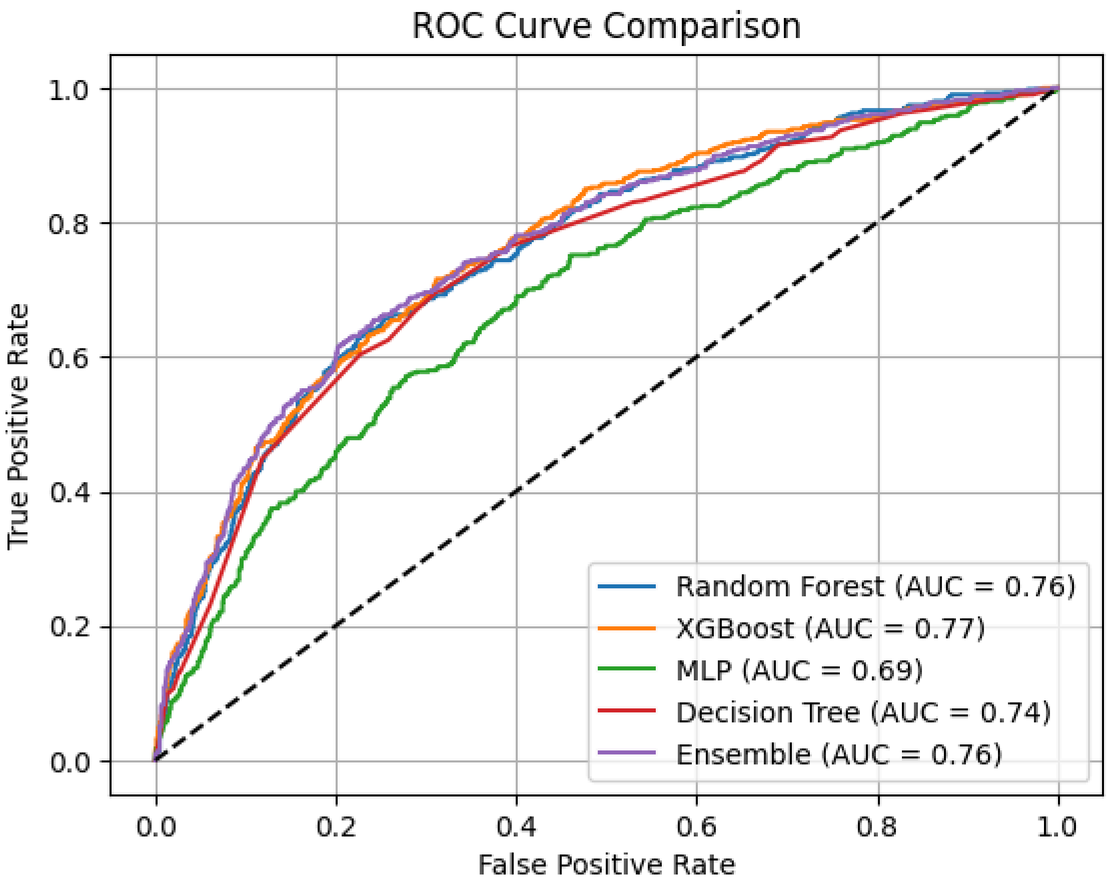
```
    metrics['Accuracy'].append(accuracy_score(y_test, y_pred))


df_metrics = pd.DataFrame(metrics).set_index("Model")

ax = df_metrics.plot(kind='bar', figsize=(10, 6), colormap='Set2')

plt.title("Model Performance Metrics on Test Set")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.xticks(rotation=15)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()

for container in ax.containers:
    ax.bar_label(container, fmt='%.2f', padding=2, fontsize=9)
plt.savefig('/content/performance_comp.png', dpi=300)
plt.show()
```



In [ ]:

```
from sklearn.metrics import roc_curve, auc

for name, model in models.items():
    y_score = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_score)
    auc_score = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {auc_score:.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
```

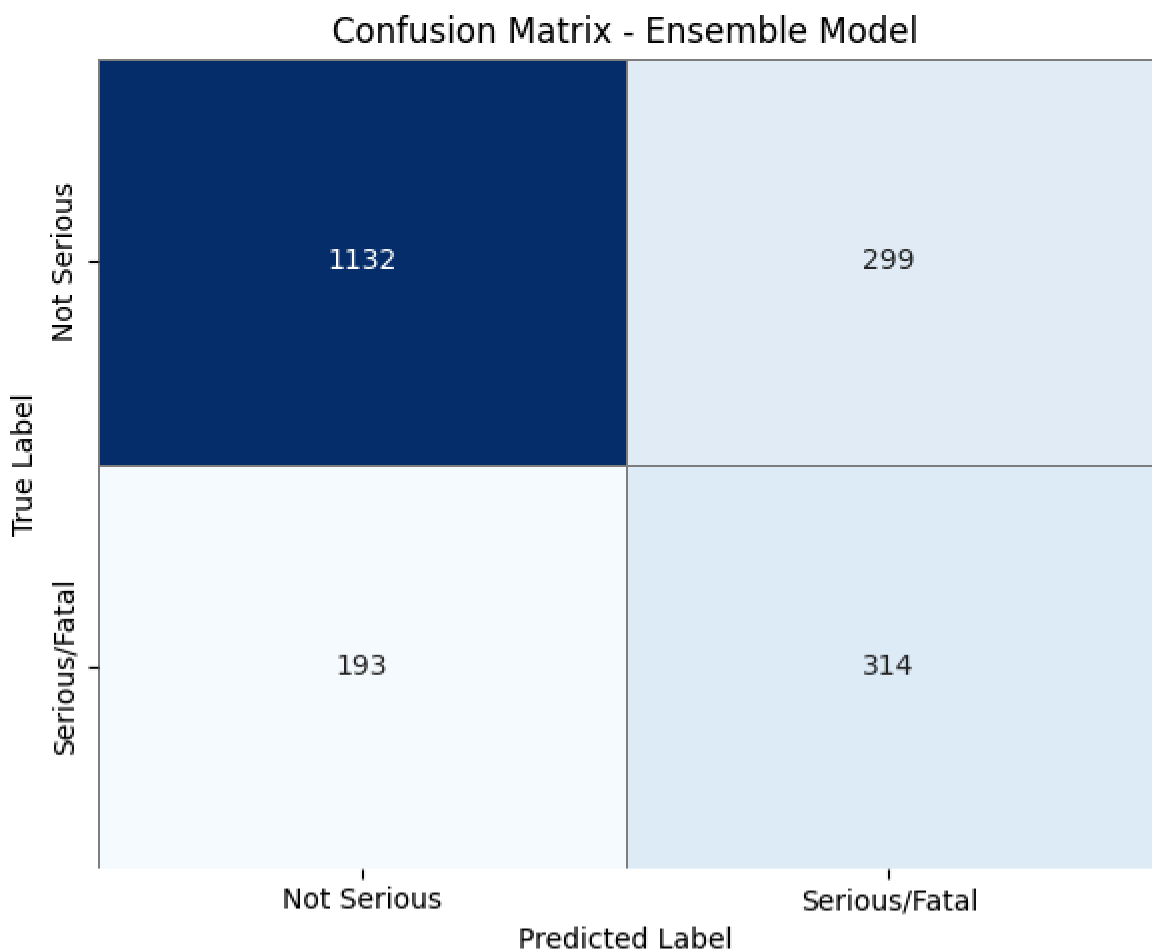
```
plt.grid()
plt.savefig('/content/roc_comp.png', dpi=300)
plt.show()
```



In [ ]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

def plot_confusion_matrix(model, X_test, y_test, labels=["Not Serious", "Serious/Fatal"], title="Confusion Matrix"):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)

    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=labels, yticklabels=labels,
                cbar=False, linewidths=0.5, linecolor='gray')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.title(title)
    plt.tight_layout()
```

```
    plt.show()

plot_confusion_matrix(ensemble, X_test, y_test, title="Confusion Matrix - Ensemble Model")
```



Model Interpretation

In [39]:

```
import numpy as np

def calculate_marginal_effect(rf_model, X_train, variable, base_class_col, target_class_col):
    X_base = X_train.copy()
    X_target = X_train.copy()

    X_base[variable] = 0  # Reset all one-hot columns for the variable
    X_base[base_class_col] = 1  # Set the base class to 1 for all rows

    X_target[variable] = 0  # Reset all one-hot columns for the variable
```

```python
    X_target[target_class_col] = 1  # Set the target class to 1 for all rows

    base_class_probs = rf_model.predict_proba(X_base)[:, 1]
    target_class_probs = rf_model.predict_proba(X_target)[:, 1]

    base_class_avg = np.mean(base_class_probs)
    target_class_avg = np.mean(target_class_probs)

    marginal_effect = target_class_avg - base_class_avg

    return marginal_effect
```

In [40]:
```python
def calculate_marginal_effects_for_variable(rf_model, X_train, variable_name, onehot_columns, base_class_col):

    results = []

    for target_class_col in onehot_columns:
        if target_class_col != base_class_col:

            marginal_effect = calculate_marginal_effect(
                rf_model,
                X_train=X_train,
                variable=onehot_columns,
                base_class_col=base_class_col,
                target_class_col=target_class_col
            )

            results.append({
                'variable': variable_name,
                'base_class': base_class_col,
                'target_class': target_class_col,
                'marginal_effect': marginal_effect
            })
    return pd.DataFrame(results)
```

In [35]:
```python
X_train_res.columns
```

```
Out[35]: Index(['CrashType_Pedestrian Failed to Yield', 'CrashType_others',
       'CrashType_pedestrian dash', 'DrvrAlcFlg_No', 'DrvrAlcFlg_Others',
       'DrvrAlcFlg_Yes', 'DrvrVehTyp_Others', 'DrvrVehTyp_Passenger Car',
       'DrvrVehTyp_Pickup', 'DrvrVehTyp_Sport Utility',
       'LightCond_Dark - Lighted Roadway',
       'LightCond_Dark - Roadway Not Lighted', 'LightCond_Dawn-Dusk',
       'LightCond_Daylight', 'LightCond_Others', 'NumLanes_2', 'NumLanes_3',
       'NumLanes_4', 'NumLanes_5', 'NumLanes_6', 'NumLanes_Others',
       'PedAgeGrp_16 to 19', 'PedAgeGrp_20 to 29', 'PedAgeGrp_30 to 39',
       'PedAgeGrp_40 to 49', 'PedAgeGrp_50 to 59', 'PedAgeGrp_above 60',
       'PedAgeGrp_below 16', 'PedAlcFlag_No', 'PedAlcFlag_Others',
       'PedAlcFlag_Yes', 'PedPos_Crosswalk Area', 'PedPos_Others',
       'PedPos_Paved Shoulder / Bike Lane / Parking Lane',
       'PedPos_Sidewalk / Shared Use Path / Driveway Crossing',
       'PedPos_Travel Lane', 'PedSex_Female', 'PedSex_not female',
       'RdCharacte_Curve - Grade', 'RdCharacte_Curve - Level',
       'RdCharacte_Others', 'RdCharacte_Straight - Grade',
       'RdCharacte_Straight - Hillcrest', 'RdCharacte_Straight - Level',
       'SpeedLimit_30 - 35 MPH', 'SpeedLimit_40 - 45 MPH',
       'SpeedLimit_50 - 55 MPH', 'SpeedLimit_60 - 75 MPH',
       'SpeedLimit_Below 25 MPH',
       'TraffCntrl_Double Yellow Line, No Passing Zone',
       'TraffCntrl_No Control Present', 'TraffCntrl_Others',
       'TraffCntrl_Stop And Go Signal', 'TraffCntrl_Stop Sign'],
      dtype='object')
```

```
In [44]: crash_type_columns = ['CrashType_Pedestrian Failed to Yield', 'CrashType_others', 'CrashType_pedestrian dash']
base_class_col = 'CrashType_pedestrian dash'
```

```
In [45]: crash_type_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='crash_type',
    onehot_columns= crash_type_columns,
    base_class_col=base_class_col
)
```

```
In [46]: crash_type_marginal_effects
```

Out[46]:

	variable	base_class	target_class	marginal_effect
0	crash_type	CrashType_pedestrian dash	CrashType_Pedestrian Failed to Yield	0.004461
1	crash_type	CrashType_pedestrian dash	CrashType_others	-0.022959

In [47]:
```
drvalc_columns = ['DrvrAlcFlg_No', 'DrvrAlcFlg_Others','DrvrAlcFlg_Yes']
base_class_col = 'DrvrAlcFlg_No'
```

In [49]:
```
drvalc_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='driver_alc',
    onehot_columns= drvalc_columns,
    base_class_col=base_class_col
)
```

In [50]:
```
drvalc_marginal_effects
```

Out[50]:

	variable	base_class	target_class	marginal_effect
0	driver_alc	DrvrAlcFlg_No	DrvrAlcFlg_Others	-0.082466
1	driver_alc	DrvrAlcFlg_No	DrvrAlcFlg_Yes	0.049052

In [51]:
```
drvveh_columns = ['DrvrVehTyp_Others', 'DrvrVehTyp_Passenger Car','DrvrVehTyp_Pickup', 'DrvrVehTyp_Sport Utility']
base_class_col = 'DrvrVehTyp_Passenger Car'
```

In [54]:
```
drvveh_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='driver_veh',
    onehot_columns= drvveh_columns,
    base_class_col=base_class_col
)
```

In [55]:
```
drvveh_marginal_effects
```

Out[55]:

	variable	base_class	target_class	marginal_effect
0	driver_veh	DrvrVehTyp_Passenger Car	DrvrVehTyp_Others	0.005124
1	driver_veh	DrvrVehTyp_Passenger Car	DrvrVehTyp_Pickup	0.008579
2	driver_veh	DrvrVehTyp_Passenger Car	DrvrVehTyp_Sport Utility	0.010697

In [56]:
```
pedalc_columns = ['PedAlcFlag_No', 'PedAlcFlag_Others','PedAlcFlag_Yes',]
base_class_col = 'PedAlcFlag_åNo'
```

In [57]:
```
pedalc_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='ped_alc',
    onehot_columns= pedalc_columns,
    base_class_col=base_class_col
)
```

In [58]:
```
pedalc_marginal_effects
```

Out[58]:

	variable	base_class	target_class	marginal_effect
0	ped_alc	PedAlcFlag_No	PedAlcFlag_Others	0.125201
1	ped_alc	PedAlcFlag_No	PedAlcFlag_Yes	0.046688

In [59]:
```
pedpos_columns = ['PedPos_Crosswalk Area', 'PedPos_Others', 'PedPos_Paved Shoulder / Bike Lane / Parking Lane','PedPos_
       'PedPos_Travel Lane']
base_class_col = 'PedPos_Travel Lane'
```

In [60]:
```
pedpos_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='ped_po',
    onehot_columns= pedpos_columns,
    base_class_col=base_class_col
)
```

In [61]:
```
pedpos_marginal_effects
```

Out[61]:

	variable	base_class	target_class	marginal_effect
0	ped_po	PedPos_Travel Lane	PedPos_Crosswalk Area	-0.078514
1	ped_po	PedPos_Travel Lane	PedPos_Others	-0.048165
2	ped_po	PedPos_Travel Lane	PedPos_Paved Shoulder / Bike Lane / Parking Lane	-0.044840
3	ped_po	PedPos_Travel Lane	PedPos_Sidewalk / Shared Use Path / Driveway C...	-0.020743

In [62]:
```
licon_columns = ['LightCond_Dark - Lighted Roadway',
       'LightCond_Dark - Roadway Not Lighted', 'LightCond_Dawn-Dusk',
       'LightCond_Daylight', 'LightCond_Others']
base_class_col = 'LightCond_Dark - Lighted Roadway'
```

In [64]:
```
licon_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='licon',
    onehot_columns= licon_columns,
    base_class_col=base_class_col
)
```

In [65]:
```
licon_marginal_effects
```

Out[65]:

	variable	base_class	target_class	marginal_effect
0	licon	LightCond_Dark - Lighted Roadway	LightCond_Dark - Roadway Not Lighted	0.043789
1	licon	LightCond_Dark - Lighted Roadway	LightCond_Dawn-Dusk	-0.044322
2	licon	LightCond_Dark - Lighted Roadway	LightCond_Daylight	-0.137908
3	licon	LightCond_Dark - Lighted Roadway	LightCond_Others	0.052232

In [67]:
```
road_columns = ['RdCharacte_Curve - Grade', 'RdCharacte_Curve - Level',
       'RdCharacte_Others', 'RdCharacte_Straight - Grade',
       'RdCharacte_Straight - Hillcrest', 'RdCharacte_Straight - Level']
base_class_col = 'RdCharacte_Straight - Level'
```

In [68]:
```
road_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='road',
    onehot_columns= road_columns,
    base_class_col=base_class_col
)
```

In [69]:
```
road_marginal_effects
```

Out[69]:

	variable	base_class	target_class	marginal_effect
0	road	RdCharacte_Straight - Level	RdCharacte_Curve - Grade	0.037993
1	road	RdCharacte_Straight - Level	RdCharacte_Curve - Level	0.036809
2	road	RdCharacte_Straight - Level	RdCharacte_Others	0.028195
3	road	RdCharacte_Straight - Level	RdCharacte_Straight - Grade	0.028753
4	road	RdCharacte_Straight - Level	RdCharacte_Straight - Hillcrest	0.025391

In [70]:
```
speed_columns = ['SpeedLimit_30 - 35 MPH', 'SpeedLimit_40 - 45 MPH',
       'SpeedLimit_50 - 55 MPH', 'SpeedLimit_60 - 75 MPH',
       'SpeedLimit_Below 25 MPH',]
base_class_col = 'SpeedLimit_Below 25 MPH'
```

In [71]:
```
speed_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='speed',
    onehot_columns= speed_columns,
    base_class_col=base_class_col
)
```

In [72]:
```
speed_marginal_effects
```

Out[72]:

	variable	base_class	target_class	marginal_effect
0	speed	SpeedLimit_Below 25 MPH	SpeedLimit_30 - 35 MPH	0.163956
1	speed	SpeedLimit_Below 25 MPH	SpeedLimit_40 - 45 MPH	0.301399
2	speed	SpeedLimit_Below 25 MPH	SpeedLimit_50 - 55 MPH	0.392900
3	speed	SpeedLimit_Below 25 MPH	SpeedLimit_60 - 75 MPH	0.522365

In [73]:
```
tractr_columns = [ 'TraffCntrl_Double Yellow Line, No Passing Zone',
       'TraffCntrl_No Control Present', 'TraffCntrl_Others',
       'TraffCntrl_Stop And Go Signal', 'TraffCntrl_Stop Sign']
base_class_col = 'TraffCntrl_No Control Present'
```

In [74]:
```
tractr_marginal_effects = calculate_marginal_effects_for_variable(
    ensemble,
    X_train_res,
    variable_name='traffic_control',
```

```
    onehot_columns= tractr_columns,
    base_class_col=base_class_col
)
```

In [75]:
```
tractr_marginal_effects
```

Out[75]:

	variable	base_class	target_class	marginal_effect
0	traffic_control	TraffCntrl_No Control Present	TraffCntrl_Double Yellow Line, No Passing Zone	0.024514
1	traffic_control	TraffCntrl_No Control Present	TraffCntrl_Others	-0.000976
2	traffic_control	TraffCntrl_No Control Present	TraffCntrl_Stop And Go Signal	-0.020117
3	traffic_control	TraffCntrl_No Control Present	TraffCntrl_Stop Sign	-0.026284

Multiclass

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, df['PedInjuryLabel'], stratify=df['PedInjuryLabel'], test_size=0.2, random_state=42
)

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
```

In [77]:
```
y_train_res.value_counts()
```

Out[77]:

PedInjuryLabel	count
4	3014
1	3014
3	3014
2	3014
0	3014

dtype: int64

In [78]:
```
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n====== {model_name} ======")
    print("Classification Report:")
    print(classification_report(y_true, y_pred, target_names=[
        "No Injury (O)", "Possible Injury (C)", "Minor Injury (B)",
        "Serious Injury (A)", "Fatal Injury (K)"
    ]))
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
```

In [79]:
```
rf = RandomForestClassifier(n_estimators=200, random_state=42)
rf.fit(X_train_res, y_train_res)
rf_pred = rf.predict(X_test)
```

In [ ]:
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'class_weight': ['balanced']
}

rf_search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_distributions=param_grid_rf,
    n_iter=5,
    scoring='f1_macro',
    cv=3,
    verbose=1,
    n_jobs=-1
)

rf_search.fit(X_train_res, y_train_res)
rf_best = rf_search.best_estimator_
```

In [ ]:
```
y_pred_rf = rf_best.predict(X_test)
print("Classification Report (Random Forest w/ SMOTE):")
print(classification_report(y_test, y_pred_rf, target_names=[
    "No Injury (O)", "Possible Injury (C)", "Minor Injury (B)",
    "Serious Injury (A)", "Fatal Injury (K)"
]))
```

```
Classification Report (Random Forest w/ SMOTE):
                     precision    recall  f1-score   support

      No Injury (O)       0.06      0.89      0.12        95
Possible Injury (C)       0.09      0.09      0.09       583
   Minor Injury (B)       0.00      0.00      0.00       753
 Serious Injury (A)       0.00      0.00      0.00       278
   Fatal Injury (K)       0.00      0.00      0.00       229

           accuracy                           0.07      1938
          macro avg       0.03      0.20      0.04      1938
       weighted avg       0.03      0.07      0.03      1938
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is i
ll-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is i
ll-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is i
ll-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [ ]:
```
evaluate_model(y_test, rf_pred, "Random Forest")
```

In [80]:
```
# XGBoost
xgb = XGBClassifier(objective='multi:softmax', num_class=5, eval_metric='mlogloss',
                    use_label_encoder=False, n_estimators=100, random_state=42)
xgb.fit(X_train_res, y_train_res)
xgb_pred = xgb.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [20:44:42] WARNING: /workspace/src/learner.c
c:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

In [ ]:
```
# MLP
# mlp = MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=500, random_state=42)
# mlp.fit(X_train_res, y_train_res)
# mlp_pred = mlp.predict(X_test)
```

In [ ]:
```
evaluate_model(y_test, rf_pred, "Random Forest")
```

In [81]:
```
evaluate_model(y_test, xgb_pred, "XGBoost")
```

```
====== XGBoost ======
Classification Report:
                     precision    recall  f1-score   support

      No Injury (O)       0.00      0.00      0.00        95
Possible Injury (C)       0.37      0.40      0.38       583
   Minor Injury (B)       0.41      0.47      0.44       753
 Serious Injury (A)       0.27      0.18      0.22       278
   Fatal Injury (K)       0.45      0.48      0.46       229

           accuracy                           0.38      1938
          macro avg       0.30      0.30      0.30      1938
       weighted avg       0.36      0.38      0.37      1938

Confusion Matrix:
[[  0  40  44   6   5]
 [  9 233 270  41  30]
 [  8 277 352  58  58]
 [  2  56 131  50  39]
 [  0  22  66  32 109]]
```

In [ ]:
```
#evaluate_model(y_test, mlp_pred, "Neural Network (MLP)")
```